pdo

Controller test
Controller Fault Handler
Power-Up Handler

Tasks

P200ms_p10
- **GUI**
  - **Handle_GUI**
- **ValveProfile**
  - **Main**

P_Sample_p5

Get raw analog input and control of the breathing - MUST HAVE HIGHEST priority of all peridoc tasks

- **measArray**
  - **Main**
  - **FirstScan**
  - **measArrays**
- **ISR_ValveCtrl**
  - **Main**

P_Sample_p8
- **Setup**
  - **Main**
  - **InitGUI**
  - **InitISR_RS232**
  - **InitVariables**
- **calcBreath**
  - **Main**
- **initVent**
  - **Main**
  - **initISR_ValveCtrl**
  - **Init_ISR_PressWatchID**
- **BreathDetect**
  - **Main**
- **measCalc**
  - **Main**
- **MainFlow**
  - **Main**

Unscheduled

Motion Groups

Ungrouped Axes
- **DymmyMaster**
- **DymmySlave**

Add-On Instructions
- **AOI_ARRAYSHIFT**
  - **Logic**
- **AOI_Ceil**
  - **Logic**
- **AOI_FillArrayInt**
  - **Logic**
- **AOI_FillArrayReal**
  - **Logic**
- **AOI_FillArrayReal50**
  - **Logic**
- **AOI_FillArray_Real2D**
  - **Logic**
- **AOI_FilterSignal**
  - **Logic**
  - **Prescan**
- **AOI_Floor**
  - **Logic**
- **AOI_Max**
  - **Logic**

- **Prescan**
- **AOI_Mean2D**
  - **Logic**
  - **Prescan**
- **AOI_Mean50**
  - **Logic**
  - **Prescan**
- **AOI_Round**
  - **Logic**
- **AOI_Trunc**
  - **Logic**

Data Types

- **User-Defined**
- **Strings**
- **Add-On-Defined**
  - **AOI_ARRAYSHIFT**
  - **AOI_Ceil**
  - **AOI_FillArrayInt**
  - **AOI_FillArrayReal**
  - **AOI_FillArrayReal50**
  - **AOI_FillArray_Real2D**
  - **AOI_FilterSignal**
  - **AOI_Floor**
  - **AOI_Max**
  - **AOI_Mean2D**
  - **AOI_Mean50**
  - **AOI_Round**
  - **AOI_Trunc**
- **Module-Defined**
- **AB:1769_HSC1_Range:C:0**
- **AB:1769_IF4FXOF2F:C:0**
- **AB:1769_IF4FXOF2F:I:0**
- **AB:1769_IF4FXOF2F:O:0**
- **AB:Embedded_AnalogIO1:C:0**
- **AB:Embedded_AnalogIO1:I:0**
- **AB:Embedded_AnalogIO1:O:0**
- **AB:Embedded_DiscreteIO1:C:0**
- **AB:Embedded_DiscreteIO1:I:0**
- **AB:Embedded_DiscreteIO1:O:0**
- **AB:Embedded_DiscreteIO:C:0**
- **AB:Embedded_DiscreteIO:I:0**
- **AB:Embedded_DiscreteIO:O:0**
- **AB:Embedded_HSC1:C:0**
- **AB:Embedded_HSC1:I:0**
- **AB:Embedded_HSC1:O:0**
- **AB:Embedded_HSC1_STRUCT_OUT1:O:0**

Trends

- **CalcBreath**
- **Test**

I/O Configuration

- **1769 Bus**
- **[0] 1769-L27ERM-QBFC1B test**
  - **[1] Embedded Discrete_IO**
  - **[2] Embedded Analog_IO**
  - **[3] Embedded Counters**
- **Ethernet**
- **1769-L27ERM-QBFC1B test**

General			
Vendor:	Rockwell Automation/Allen-Bradley	Mode:	Offline
Revision:	32.013	Key Switch Position:	Offline
Chassis Type:	<none>	Created:	26-03-2020 15:02:52
Slot:	0	Edited:	02-04-2020 20:43:03
Date/Time			
Date and Time:	<offline>		
Time Zone:	<offline>		
Daylight Saving (+00:00):	<offline>		
Enable Time Synchronization:	No		
Is the system time master:	<offline>		
Is a synchronized time slave:	<offline>		
Duplicate CST Master Detected:	<offline>		
CST Mastership disabled:	<offline>		
No CST Master:	<offline>		
Advanced Time Sync			
CIP Sync Time Synchronization:	Disabled		
Advanced			
Controller Fault Handler:	<none>	Match Project To Controller:	No
Power-Up Handler:	<none>	Serial Number:	C01FB861
System Overhead Time Slice:	20 %	Allow Consumed Tags to Use RPI Provided by Producer:	No
During unused System Overhead Time Slice:	Run Continuous Task		
SFC Execution			
Execution Control:	Execute current active steps only	Last Scan of Active Step:	Don't scan
Restart Position:	Restart at most recently executed step		
Nonvolatile Memory			
<offline>			
Memory (Estimate)			
Memory Option:	1769-L27ERM-QBFC1B		
Estimated I/O Memory			
Total Memory:	1,048,576 bytes	Max Used:	4,640 bytes
Free Memory:	1,043,936 bytes	Largest Block Free:	1,043,936 bytes
Used Memory:	4,640 bytes		
Estimated Data and Logic Memory			
Total Memory:	1,048,576 bytes	Max Used:	665,744 bytes
Free Memory:	382,832 bytes	Largest Block Free:	382,832 bytes
Used Memory:	665,744 bytes		
Security:			
Primary Security Authority:	No Protection	Restrict Communications Except Through Selected Slots:	No
Use only the selected Security Authority for Authentication and Authorization:	No	Selected Slots:	
Secondary Security Authority:	No Protection	Changes To Detect:	16#ffff_ffff_ffff_ffff
Use only the selected Security Authority for Authentication and Authorization:	No	Audit Value:	<offline>
Permission Set:			
Port Configuration			
Port 1			
Enable:	Yes		
Port 2			
Enable:	Yes		

Name	Value	Data Type	Scope
_CopDes1	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopDes1 - ISR_ValveCtrl/Main - #142, #143, #144, *#140*			
_CopDes4	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopDes4 - ISR_ValveCtrl/Main - #163, #164, #165, #168, *#161*			
_CopDes5	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopDes5 - ISR_ValveCtrl/Main - #170, #171, #172, *#168*			
_CopDes6	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopDes6 - ISR_ValveCtrl/Main - #179, #180, #181, *#177*			
_CopDes8	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopDes8 - ISR_ValveCtrl/Main - #198, #199, #200, *#196*			
_CopLength	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopLength - ISR_ValveCtrl/Main - #135, #136, #137, #140, *#134*			
_CopLength1	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopLength1 - ISR_ValveCtrl/Main - #142, #143, #144, *#141*			
_CopLength3	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopLength3 - ISR_ValveCtrl/Main - #150, #151, #152, *#149*			
_CopLength4	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopLength4 - ISR_ValveCtrl/Main - #163, #164, #165, *#162*			
_CopLength5	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopLength5 - ISR_ValveCtrl/Main - #170, #171, #172, *#169*			
_CopLength6	0	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopLength6 - ISR_ValveCtrl/Main - #179, #180, #181, *#178*			
_CopLength7	2060	DINT	test
Constant	No		
External Access:	Read/Write		
*_CopLength7 - ISR_ValveCtrl/Main - #190, #191, #192, *#189*			
_CopLength8	0	DINT	test
Constant	No		
External Access:	Read/Write		

_CopLength8 (Continued)
*_CopLength8 - ISR_ValveCtrl/Main - #198, #199, #200, *#197*

_CopSource 0 DINT test
Constant No
External Access: Read/Write
*_CopSource - ISR_ValveCtrl/Main - #135, #136, #137, *#133*

_CopSource1 0 DINT test
Constant No
External Access: Read/Write
*_CopSource1 - ISR_ValveCtrl/Main - #142, #143, #144, *#139*

_CopSource3 0 DINT test
Constant No
External Access: Read/Write
*_CopSource3 - ISR_ValveCtrl/Main - #150, #151, #152, *#148*

_CopSource4 0 DINT test
Constant No
External Access: Read/Write
*_CopSource4 - ISR_ValveCtrl/Main - #163, #164, #165, *#160*

_CopSource5 0 DINT test
Constant No
External Access: Read/Write
*_CopSource5 - ISR_ValveCtrl/Main - #170, #171, #172, *#167*

_CopSource6 0 DINT test
Constant No
External Access: Read/Write
*_CopSource6 - ISR_ValveCtrl/Main - #179, #180, #181, *#176*

_CopSource7 -30 DINT test
Constant No
External Access: Read/Write
*_CopSource7 - ISR_ValveCtrl/Main - #190, #191, #192, *#188*

_CopSource8 0 DINT test
Constant No
External Access: Read/Write
*_CopSource8 - ISR_ValveCtrl/Main - #198, #199, #200, *#195*

_edbAoiFilterFlowAir AOI_FilterSignal test
Constant No
External Access: Read/Write
*_edbAoiFilterFlowAir - measArray/measArrays - *#109*
_edbAoiFilterFlowAir.EnableIn 1 BOOL
Enable Input - System Defined Parameter
_edbAoiFilterFlowAir.EnableOut 1 BOOL
Enable Output - System Defined Parameter
_edbAoiFilterFlowAir.In_bInitialize
0 BOOL
*_edbAoiFilterFlowAir.In_bInitialize - measArray/measArrays - *#101*
_edbAoiFilterFlowAir.Par_eType 0 DINT
0: MEDIAN; 1: MEAN
*_edbAoiFilterFlowAir.Par_eType - measArray/measArrays - *#84*
_edbAoiFilterFlowAir.Par_nWindowLeft
5 DINT
*_edbAoiFilterFlowAir.Par_nWindowLeft - measArray/measArrays - *#82*
_edbAoiFilterFlowAir.Par_nWindowsRight
5 DINT
*_edbAoiFilterFlowAir.Par_nWindowsRight - measArray/measArrays - *#83*
_edbAoiFilterFlowAir.Par_bUseWeights

_edbAoiFilterFlowAir (Continued)
0 BOOL
Used for mean filtering
*_edbAoiFilterFlowAir.Par_bUseWeights - measArray/measArrays - *#85*
_edbAoiFilterFlowAir.Out_Buffering
0 BOOL
_edbAoiFilterFlowAir.Out_Buffering - measArray/measArrays - #116
_edbAoiFilterFlowAir.Out_rFilteredSignalDelayed
980.0 REAL
_edbAoiFilterFlowAir.Out_rFilteredSignalDelayed - measArray/measArrays - #122

_edbAoiFilterFlowO2 AOI_FilterSignal test
Constant No
External Access: Read/Write
*_edbAoiFilterFlowO2 - measArray/measArrays - *#111*
_edbAoiFilterFlowO2.EnableIn 1 BOOL
Enable Input - System Defined Parameter
_edbAoiFilterFlowO2.EnableOut 1 BOOL
Enable Output - System Defined Parameter
_edbAoiFilterFlowO2.In_bInitialize
0 BOOL
*_edbAoiFilterFlowO2.In_bInitialize - measArray/measArrays - *#102*
_edbAoiFilterFlowO2.Par_eType 0 DINT
0: MEDIAN; 1: MEAN
*_edbAoiFilterFlowO2.Par_eType - measArray/measArrays - *#89*
_edbAoiFilterFlowO2.Par_nWindowLeft
5 DINT
*_edbAoiFilterFlowO2.Par_nWindowLeft - measArray/measArrays - *#87*
_edbAoiFilterFlowO2.Par_nWindowsRight
5 DINT
*_edbAoiFilterFlowO2.Par_nWindowsRight - measArray/measArrays - *#88*
_edbAoiFilterFlowO2.Par_bUseWeights
0 BOOL
Used for mean filtering
*_edbAoiFilterFlowO2.Par_bUseWeights - measArray/measArrays - *#90*
_edbAoiFilterFlowO2.Out_Buffering
0 BOOL
_edbAoiFilterFlowO2.Out_Buffering - measArray/measArrays - #117
_edbAoiFilterFlowO2.Out_rFilteredSignalDelayed
981.0 REAL
_edbAoiFilterFlowO2.Out_rFilteredSignalDelayed - measArray/measArrays - #126

_edbAoiFilterPress AOI_FilterSignal test
Constant No
External Access: Read/Write
*_edbAoiFilterPress - measArray/measArrays - *#110*
_edbAoiFilterPress.EnableIn 1 BOOL
Enable Input - System Defined Parameter
_edbAoiFilterPress.EnableOut 1 BOOL
Enable Output - System Defined Parameter
_edbAoiFilterPress.In_bInitialize 0 BOOL
*_edbAoiFilterPress.In_bInitialize - measArray/measArrays - *#103*
_edbAoiFilterPress.Par_eType 0 DINT
0: MEDIAN; 1: MEAN
*_edbAoiFilterPress.Par_eType - measArray/measArrays - *#94*
_edbAoiFilterPress.Par_nWindowLeft
5 DINT
*_edbAoiFilterPress.Par_nWindowLeft - measArray/measArrays - *#92*
_edbAoiFilterPress.Par_nWindowsRight
5 DINT
*_edbAoiFilterPress.Par_nWindowsRight - measArray/measArrays - *#93*
_edbAoiFilterPress.Par_bUseWeights
0 BOOL
Used for mean filtering

_edbAoiFilterPress (Continued)
*_edbAoiFilterPress.Par_bUseWeights - measArray/measArrays - *#95*
_edbAoiFilterPress.Out_Buffering 0 BOOL
_edbAoiFilterPress.Out_Buffering - measArray/measArrays - #118
_edbAoiFilterPress.Out_rFilteredSignalDelayed
3947.0 REAL
_edbAoiFilterPress.Out_rFilteredSignalDelayed - measArray/measArrays - #124

_edbCanStore 1 BOOL test
Constant No
External Access: Read/Write
*_edbCanStore - measArray/measArrays - #120, *#116*

_edbFlowArrFiltered REAL[1000,2] test
Constant No
External Access: Read/Write
*_edbFlowArrFiltered - MainFlow/Main - *#62*
*_edbFlowArrFiltered - measArray/measArrays - *#122, *#123*

_edbFlowO2Filtered REAL[1000,2] test
Constant No
External Access: Read/Write
*_edbFlowO2Filtered - MainFlow/Main - *#63*
*_edbFlowO2Filtered - measArray/measArrays - *#124, *#125*

_edbHalfWindowFilterSize 5 DINT test
Constant No
External Access: Read/Write
*_edbHalfWindowFilterSize - measArray/measArrays - #119, #121, #82, #83, #87, #88, #92, #93, *#67*

_edbIdx 590 DINT test
Constant No
External Access: Read/Write
*_edbIdx - measArray/measArrays - #122, #123, #124, #125, #126, #127, *#121*

_edbParUseWeigths 0 BOOL test
Constant No
External Access: Read/Write
*_edbParUseWeigths - measArray/measArrays - #85, #90, #95, *#74*

_edbParWeights REAL[31] test
Constant No
External Access: Read/Write
*_edbParWeights - measArray/measArrays - *#109, *#110, *#111*

_edbPressArrFiltered REAL[1000,2] test
Constant No
External Access: Read/Write
*_edbPressArrFiltered - MainFlow/Main - *#64*
*_edbPressArrFiltered - measArray/measArrays - *#126, *#127*

_edbTypeOfFilter 0 DINT test
Constant No
External Access: Read/Write
*_edbTypeOfFilter - measArray/measArrays - #84, #89, #94, *#73*

_endPosition 0 DINT test
Constant No
External Access: Read/Write
*_endPosition - ISR_ValveCtrl/Main - #104, #106, #121, #125, #127, #156, #194, *#105, *#121*
*_endPosition - measArray/FirstScan - *#107*

_endPositionOld 0 DINT test
Constant No

_endPositionOld (Continued)
External Access: Read/Write
*_endPositionOld - ISR_ValveCtrl/Main - #104, *#106, *#205*

_EndSendPosition 0 DINT test
Constant No
External Access: Read/Write
*_EndSendPosition - ISR_ValveCtrl/Main - #158, #169, #178, #197, *#156, *#194*

_i 1000 DINT test
Constant No
External Access: Read/Write
*_i - measCalc/Main - #12, #18, #18, #19, #27, #27, #29, #30, #32, #41, #41, #42, *#10*

_J 196 INT test
Constant No
External Access: Read/Write
*_J - measCalc/Main - #108, *#107*

_n 412 DINT test
Constant No
External Access: Read/Write
*_n - measCalc/Main - *#139*

_PointAfterEnd 30 DINT test
Constant No
External Access: Read/Write
*_PointAfterEnd - ISR_ValveCtrl/Main - #122, #125, #156, #194, *#122*
*_PointAfterEnd - measArray/FirstScan - *#108*

_StartPointBerfore 30 DINT test
Constant No
External Access: Read/Write
*_StartPointBerfore - ISR_ValveCtrl/Main - #119, #125, #129, #149, #161, #177, #187, *#119*
*_StartPointBerfore - measArray/FirstScan - *#105*

_StartPosition 0 DINT test
Constant No
External Access: Read/Write
*_StartPosition - ISR_ValveCtrl/Main - #120, #125, #127, #129, #160, #162, #168, #176, #178, #187, #28, #30, *#120, *#29*
*_StartPosition - measArray/FirstScan - *#106*

_StartpositionOld 0 DINT test
Constant No
External Access: Read/Write
*_StartpositionOld - ISR_ValveCtrl/Main - #28, *#206, *#30*

_T 0 DINT test
Constant No
External Access: Read/Write
*_T - measCalc/Main - #149, #149, #150, #150, #151, #151, #152, #152, #153, #153, #154, #154, #155, #155, #156, #156, #157, #157, #158, #158, #159, #159, *#147*

_z 51 DINT test
Constant No
External Access: Read/Write
*_z - measCalc/Main - #183, #183, #187, #188, #194, #195, #198, *#182, *#185, *#190, *#199*

AirConstantFlow 68.35443 REAL test
Constant No
External Access: Read/Write
*AirConstantFlow - calcBreath/Main - #46, #66, *#41*

AirDown CAM_PROFILE[1010] test

AirDown (Continued)
Constant No
External Access: Read/Write
AirDown - ValveProfile/Main - 0(MAPC)

AirmADer 0.0 REAL test
Constant No
External Access: Read/Write
*AirmADer - ISR_ValveCtrl/Main - *#63*

AirmAOut 0.0 REAL test
Constant No
External Access: Read/Write
*AirmAOut - ISR_ValveCtrl/Main - #64, *#63*

AirmASlope 49211.336 REAL test
Constant No
External Access: Read/Write
*AirmASlope - ISR_ValveCtrl/Main - *#63*

AirRampFlowIncreasePerAdjust 3.4177215 REAL test
Constant No
External Access: Read/Write
*AirRampFlowIncreasePerAdjust - calcBreath/Main - #63, *#46*

AirUp CAM_PROFILE[1010] test
Constant No
External Access: Read/Write
AirUp - ISR_ValveCtrl/Main - #63
AirUp - ValveProfile/Main - 0(MAPC)

AnalogInputFlow 1059 INT test
AliasFor: Local:2:I.Ch3Data
Base Tag: Local:2:I.Ch3Data
Constant No
External Access: Read/Write
AnalogInputFlow - measArray/measArrays - #32

AnalogInputFlowAir 980 INT test
AliasFor: Local:2:I.Ch1Data
Base Tag: Local:2:I.Ch1Data
Constant No
External Access: Read/Write
AnalogInputFlowAir - measArray/measArrays - #109, #34

AnalogInputFlowO2 981 INT test
AliasFor: Local:2:I.Ch0Data
Base Tag: Local:2:I.Ch0Data
Constant No
External Access: Read/Write
AnalogInputFlowO2 - measArray/measArrays - #111, #36

AnalogInputPressure 3943 INT test
AliasFor: Local:2:I.Ch2Data
Base Tag: Local:2:I.Ch2Data
Constant No
External Access: Read/Write
AnalogInputPressure - ISR_ValveCtrl/Main - #1
AnalogInputPressure - measArray/measArrays - #110, #38

AnalogOutFlowAir 0 INT test
AliasFor: Local:2:O.Ch1Data
Base Tag: Local:2:O.Ch1Data
Constant No

AnalogOutFlowAir (Continued)
External Access: Read/Write
*AnalogOutFlowAir - ISR_ValveCtrl/Main - *#53, *#64, *#99*

AnalogOutFlowO2 0 INT test
AliasFor: Local:2:O.Ch0Data
Base Tag: Local:2:O.Ch0Data
Constant No
External Access: Read/Write
*AnalogOutFlowO2 - ISR_ValveCtrl/Main - *#52, *#62, *#98*

avgCrsDyn 0.44807863 REAL test
1 min avg of dynamic respiratory system compliance in ml/cmH2O, from measAvgCalc()
Constant No
External Access: Read/Write
*avgCrsDyn - measCalc/Main - *#221*

avgCrsQStat 0.50136733 REAL test
1 min avg of quasistatic respiratory system compliance in ml/cmH2O, from measAvgCalc()
Constant No
External Access: Read/Write
*avgCrsQStat - measCalc/Main - *#223*

avgFlowImax 18987.0 REAL test
1 min avg of max insp flow in l/min, from measAvgCalc()
Constant No
External Access: Read/Write
*avgFlowImax - measCalc/Main - *#211*

avgMV 41214.273 REAL test
1 min avg of minute ventilation in l/min, from measAvgCalc()
Constant No
External Access: Read/Write
*avgMV - measCalc/Main - *#219*

avgPEEP 4402.0 REAL test
1 min avg of positive end expiratory pressure in cmH2O, from measAvgCalc()
Constant No
External Access: Read/Write
*avgPEEP - measCalc/Main - *#209*

avgPpeak 10534.0 REAL test
1 min avg of peak insp pressure in cmH2O, from measAvgCalc()
Constant No
External Access: Read/Write
*avgPpeak - measCalc/Main - *#207*

avgPplat 9882.25 REAL test
1 min avg of plateau insp pressure in cmH2O, from measAvgCalc()
Constant No
External Access: Read/Write
*avgPplat - measCalc/Main - *#213*

avgRaw 2.059567 REAL test
1 min avg of airway resistance in cmH2O/l/s, from measAvgCalc()
Constant No
External Access: Read/Write
*avgRaw - measCalc/Main - *#225*

avgRR 15.0 REAL test
1 min avg of respiratory rate in 1/min, from measAvgCalc()
Constant No
External Access: Read/Write
*avgRR - measCalc/Main - *#215*

avgVti 2747.6182 REAL test
1 min avg of insp tidal volume in ml, from measAvgCalc()
Constant No
External Access: Read/Write
*avgVti - measCalc/Main - *#217*

BackingTag MOTION_INSTRUCTION test
Constant No
External Access: Read/Write
*BackingTag - ValveProfile/Main - *0(MAPC)*

BreathDuration 4 INT test
Constant No
External Access: Read/Write
*BreathDuration - measCalc/Main - #173, #65, *#64*

BreathDurationArr REAL[50] test
50 length array with seconds for each breath for calculation of minute avgVentStateArr. Calcualted from knowing meas at 100Hz for all breath data

Constant No
External Access: Read/Write
*BreathDurationArr - initVent/initISR_ValveCtrl - *#5*
*BreathDurationArr - measArray/FirstScan - *#52*
*BreathDurationArr - measCalc/Main - #159, #183, #188, #195, *#159*
*BreathDurationArr[0] - measCalc/Main - *#173*

BreathDurationInMS 4000.0002 REAL test
Constant No
External Access: Read/Write
*BreathDurationInMS - calcBreath/Main - #11, #12, #15, #20, #21, *#10*

CalcNewData 0 BOOL test
Constant No
External Access: Read/Write
*CalcNewData - ISR_ValveCtrl/Main - *#207, *#211*
*CalcNewData - measCalc/Main - #4, *#227*

ConstansFlowNrCalc 40.0 REAL test
Constant No
External Access: Read/Write
*ConstansFlowNrCalc - calcBreath/Main - #49, *#48*

ConstantFlowNrOfAdjustments 40.0 REAL test
Constant No
External Access: Read/Write
*ConstantFlowNrOfAdjustments - calcBreath/Main - #64, #67, #67, *#49*

ConstantFlowTimeInMS 400.0 REAL test
Constant No
External Access: Read/Write
*ConstantFlowTimeInMS - calcBreath/Main - #27, #48, *#22*

ConstantInspiratoryFlow 72.0 REAL test
Constant No
External Access: Read/Write
*ConstantInspiratoryFlow - calcBreath/Main - #31, #40, #41, *#27*

counter 594 DINT test
Counter for Flow fill array
Constant No
External Access: Read/Write
counter - ISR_ValveCtrl/Main - #105, #29
*counter - measArray/FirstScan - *#3*

counter (Continued)
*counter - measArray/measArrays - #119, #121, #132, #135, #31, #31, #32, #33, #34, #35, #36, #37, #38, #39, #58, *#132, *#136*

CrsDyn 0.5221349 REAL test
dynamic respiraotry system compliance in ml/cmH2O
Constant No
External Access: Read/Write
*CrsDyn - measCalc/Main - #170, *#127*

CrsDynArr REAL[50] test
50 length array with dynamic respiraotry system compliance in ml/cmH2O, from measCalc()
Constant No
External Access: Read/Write
*CrsDynArr - measArray/FirstScan - *#49*
*CrsDynArr - measCalc/Main - #156, *#156, *#220*
*CrsDynArr[0] - measCalc/Main - *#170*

CrsQStat 0.59738743 REAL test
quasistatic respiratory system compliance in ml/cmH2O
Constant No
External Access: Read/Write
*CrsQStat - measCalc/Main - #171, *#131, *#133*

CrsQStatArr REAL[50] test
50 length array with quasistatic respiratory system compliance in ml/cmH2O, from measCalc()
Constant No
External Access: Read/Write
*CrsQStatArr - measArray/FirstScan - *#50*
*CrsQStatArr - measCalc/Main - #157, *#157, *#222*
*CrsQStatArr[0] - measCalc/Main - *#171*

CsumData 659 DINT test
Constant No
External Access: Read/Write
*CsumData - MainFlow/Main - #11, #12, *#9*

CsumDataOld 659 DINT test
Constant No
External Access: Read/Write
*CsumDataOld - MainFlow/Main - #11, *#12*

data_changed 0 BOOL test
Constant No
External Access: Read/Write
*data_changed - GUI/Handle_GUI - #139, #194, #31, #43, #92, *#140, *#143, *#185, *#44, *#48, *#93, *#96*
*data_changed - measArray/FirstScan - *#28*

DymmyMaster AXIS_VIRTUAL test
External Access: Read/Write
DymmyMaster - ValveProfile/Main - 0(MAPC)

DymmySlave AXIS_VIRTUAL test
External Access: Read/Write
DymmySlave - ValveProfile/Main - 0(MAPC)

endI 0 DINT test
Constant No
External Access: Read/Write
*endI - measArray/FirstScan - *#39*
*endI - measCalc/Main - #206, #208, #210, #212, #214, #216, #218, #220, #222, #224, *#189, *#198*

EndPoint 194 INT test
Constant No
External Access: Read/Write

EndPoint (Continued)
*EndPoint - measCalc/Main - #99, *#98*

Exp 0 BOOL test
Constant No
External Access: Read/Write
*Exp - ISR_ValveCtrl/Main - #34, #6, #6, *#115, *#13, *#17, *#72, *#8*
*Exp - MainFlow/Main - #53, *#67*

expCount 0 INT test
between 1-1000 to indicate where valveCtrl should read from expOpenCloseCtrl[]
Constant No
External Access: Read/Write
*expCount - ISR_ValveCtrl/Main - #100, #102, #15, #7, #96, *#42, *#96*
*expCount - MainFlow/Main - #54, *#68*

expCountLength 300 INT test
between 1-1000 to indicate when valveCtrl should stop reading from expOpenCloseCtrl[], copied from nextExpCountLength by ISR_ValveCtrl()
Constant No
External Access: Read/Write
*expCountLength - initVent/Main - #11, *#9*
*expCountLength - ISR_ValveCtrl/Main - #102, #15, #7, *#224*
expCountLength - MainFlow/Main - #54

expCountlengthCalc 300.00003 REAL test
Constant No
External Access: Read/Write
*expCountlengthCalc - calcBreath/Main - #17, *#16*

expCountLengthNew 300.0 REAL test
Constant No
External Access: Read/Write
*expCountLengthNew - calcBreath/Main - #74, #83, *#17*

expEndI 0 INT test
Constant No
External Access: Read/Write
*expEndI - measArray/FirstScan - *#37*
*expEndI - measCalc/Main - #139, #37, #57, #64, *#42, *#58, *#8*

expOpen 1 BOOL test
AliasFor: Local:1:O.Data.0
Base Tag: Local:1:O.Data.0
Constant No
External Access: Read/Write
*expOpen - ISR_ValveCtrl/Main - *#100, *#54, *#60*

expOpenCloseCtrl INT[1000] test
1000 length array with open (1) or close (0) for exp valve, moved from nextexpOpenCloseCtrl[] by ISR_ValveCtrl()
Constant No
External Access: Read/Write
expOpenCloseCtrl - ISR_ValveCtrl/Main - #100
*expOpenCloseCtrl[0] - initVent/Main - *#7*
*expOpenCloseCtrl[0] - ISR_ValveCtrl/Main - *#223*

expOpenCloseCtrlNew INT[1000] test
Constant No
External Access: Read/Write
*expOpenCloseCtrlNew - calcBreath/Main - *#56, *#74*
expOpenCloseCtrlNew[0] - calcBreath/Main - #82

expStartI 0 INT test
Constant No
External Access: Read/Write

expStartI (Continued)
*expStartI - measArray/FirstScan - *#36*
*expStartI - measCalc/Main - #107, #23, #37, #53, #58, #78, #86, #97, #98, *#30, *#32, *#54, *#7*

expTimeInMS 3000.0002 REAL test
Constant No
External Access: Read/Write
*expTimeInMS - calcBreath/Main - *#15*

F1_pushed_new 0 BOOL test
Constant No
External Access: Read/Write
F1_pushed_new - GUI/Handle_GUI - #40, #42, #82

F1_pushed_old 0 BOOL test
Constant No
External Access: Read/Write
*F1_pushed_old - GUI/Handle_GUI - #40, #82, *#41, *#83*

F2_pushed_new 0 BOOL test
Constant No
External Access: Read/Write
F2_pushed_new - GUI/Handle_GUI - #129, #89, #91

F2_pushed_old 0 BOOL test
Constant No
External Access: Read/Write
*F2_pushed_old - GUI/Handle_GUI - #129, #89, *#130, *#90*

F3_pushed_new 0 BOOL test
Constant No
External Access: Read/Write
F3_pushed_new - GUI/Handle_GUI - #136, #138, #176

F3_pushed_old 0 BOOL test
Constant No
External Access: Read/Write
*F3_pushed_old - GUI/Handle_GUI - #136, #176, *#137, *#177*

Fill AOI_FillArrayReal test
Constant No
External Access: Read/Write
*Fill - calcBreath/Main - *#54, *#55*
Fill.EnableIn 1 BOOL
Enable Input - System Defined Parameter
Fill.EnableOut 1 BOOL
Enable Output - System Defined Parameter

Fill50 AOI_FillArrayReal50 test
Constant No
External Access: Read/Write
*Fill50 - initVent/initISR_ValveCtrl - *#5*
*Fill50 - measArray/FirstScan - *#42, *#43, *#44, *#45, *#46, *#47, *#48, *#49, *#50, *#51, *#52*
Fill50.EnableIn 1 BOOL
Enable Input - System Defined Parameter
Fill50.EnableOut 1 BOOL
Enable Output - System Defined Parameter

Fillint AOI_FillArrayInt test
Constant No
External Access: Read/Write
*Fillint - calcBreath/Main - *#56, *#74*
Fillint.EnableIn 1 BOOL
Enable Input - System Defined Parameter

Fillint (Continued)
Fillint.EnableOut 1 BOOL
Enable Output - System Defined Parameter

FillLoop AOI_FillArray_Real2D test
Constant No
External Access: Read/Write
*FillLoop - MainFlow/Main - *#56, *#57, *#58, *#59, *#60, *#61, *#62, *#63, *#64*
*FillLoop - measArray/FirstScan - *#8, *#9*
*FillLoop - measCalc/Main - *#228, *#229, *#230*
FillLoop.EnableIn 1 BOOL
Enable Input - System Defined Parameter
FillLoop.EnableOut 1 BOOL
Enable Output - System Defined Parameter

FiO2 25 INT test
25-100%
Constant No
External Access: Read/Write
FiO2 - calcBreath/Main - #36
*FiO2 - GUI/Handle_GUI - #145, #146, #98, #99, *#100, *#145, *#147, *#98*
FiO2 - MainFlow/Main - #14, #9
*FiO2 - measArray/FirstScan - #61, *#62, *#64*

FiO2Running 25 INT test
Constant No
External Access: Read/Write
*FiO2Running - MainFlow/Main - *#14*
FiO2Running - measArray/FirstScan - #64

Floor AOI_Floor test
Constant No
External Access: Read/Write
*Floor - calcBreath/Main - *#14, *#17, *#44, *#49, *#51*
Floor.EnableIn 1 BOOL
Enable Input - System Defined Parameter
Floor.EnableOut 1 BOOL
Enable Output - System Defined Parameter

Flow REAL[1000,2] measArray
Constant No
External Access: Read/Write
*Flow - measArray/measArrays - *#32, *#33*

FlowAir REAL[1000,2] test
1000 length 2D FIFO array with flow values and status bits indicating if values have been analysed (1) or not (0) by measCalc()
Constant No
External Access: Read/Write
FlowAir - ISR_ValveCtrl/Main - #135, #142, #150, #163, #170, #179, #190, #198
*FlowAir - MainFlow/Main - *#56*
*FlowAir - measArray/FirstScan - *#8*
*FlowAir - measArray/measArrays - *#34, *#35*

FlowArrMem REAL[1000,2] test
Constant No
External Access: Read/Write
*FlowArrMem - ISR_ValveCtrl/Main - *#142, *#163, *#170, *#179, *#198*
*FlowArrMem - MainFlow/Main - *#59*
*FlowArrMem - measCalc/Main - #108, #12, #18, #18, #27, #27, #41, #41, *#228, *#86*
*FlowArrMem[0,0] - ISR_ValveCtrl/Main - *#135, *#150, *#190*

flowCalcOn 0 BOOL test
Constant No
External Access: Read/Write

flowCalcOn (Continued)
*flowCalcOn - initVent/initISR_ValveCtrl - *#6*
flowCalcOn - ISR_ValveCtrl/Main - #117

FlowImax 18987.0 REAL test
max insp flow in l/min
Constant No
External Access: Read/Write
*FlowImax - measCalc/Main - #121, #165, *#87*

FlowImaxArr REAL[50] test
50 length array with max insp flow in l/min, from measCalc()
Constant No
External Access: Read/Write
*FlowImaxArr - measArray/FirstScan - *#44*
*FlowImaxArr - measCalc/Main - #151, *#151, *#210*
*FlowImaxArr[0] - measCalc/Main - *#165*

FlowImaxValue AOI_Max test
Constant No
External Access: Read/Write
*FlowImaxValue - measCalc/Main - *#86*
FlowImaxValue.EnableIn 0 BOOL
Enable Input - System Defined Parameter
FlowImaxValue.EnableOut 0 BOOL
Enable Output - System Defined Parameter
FlowImaxValue.Out_Max 18987.0 REAL
FlowImaxValue.Out_Max - measCalc/Main - #87

FlowO2 REAL[1000,2] test
Constant No
External Access: Read/Write
FlowO2 - ISR_ValveCtrl/Main - #136, #143, #151, #164, #171, #180, #191, #199
*FlowO2 - MainFlow/Main - *#57*
*FlowO2 - measArray/measArrays - *#36, *#37*

FlowO2Mem REAL[1000,2] test
Constant No
External Access: Read/Write
*FlowO2Mem - ISR_ValveCtrl/Main - *#143, *#164, *#171, *#180, *#199*
*FlowO2Mem - MainFlow/Main - *#60*
*FlowO2Mem - measCalc/Main - *#229*
*FlowO2Mem[0,0] - ISR_ValveCtrl/Main - *#136, *#151, *#191*

FractionAir 0.94936705 REAL test
Constant No
External Access: Read/Write
*FractionAir - calcBreath/Main - #37, #41, *#36*

FractionO2 0.050632954 REAL test
Constant No
External Access: Read/Write
*FractionO2 - calcBreath/Main - #40, *#37*

GetSlaveValue MOTION_INSTRUCTION test
Constant No
External Access: Read/Write
*GetSlaveValue - ISR_ValveCtrl/Main - *#61, *#63*

HMI_ScreenNo 0 DINT test
Constant No
External Access: Read/Write
*HMI_ScreenNo - GUI/Handle_GUI - *#12, *#3, *#7*

HMIStart 0 BOOL test
Constant No
External Access: Read/Write
HMIStart - GUI/Handle_GUI - #5

HMIStop 0 BOOL test
Constant No
External Access: Read/Write
HMIStop - GUI/Handle_GUI - #10

i 101 DINT test
Constant No
External Access: Read/Write
*i - calcBreath/Main - #60, #61, #61, #62, #62, #63, #63, #64, #64, #65, #66, #67, #67, #68, #69, *#59*
i - measCalc/Main - #189

I_E '1:3' STRING test
part of I:E ratio: 1, 2, 3 or 4
Constant No
External Access: Read/Write
*I_E - GUI/Handle_GUI - *#123, *#170*
I_E.LEN 3 DINT
part of I:E ratio: 1, 2, 3 or 4
I_E.DATA SINT
part of I:E ratio: 1, 2, 3 or 4

IE STRING[4] test
Constant No
External Access: Read/Write
IE - GUI/Handle_GUI - #123, #170

IE_counter 2 DINT test
Constant No
External Access: Read/Write
*IE_counter - GUI/Handle_GUI - #116, #117, #123, #163, #164, #170, *#116, *#118, *#163, *#165*
IE_counter - MainFlow/Main - #6

IE_E 3 INT test
Constant No
External Access: Read/Write
IE_E - calcBreath/Main - #11
*IE_E - MainFlow/Main - #18, #9, *#6*
*IE_E - measArray/FirstScan - #85, *#86, *#88*

IE_ERunning 3 INT test
Constant No
External Access: Read/Write
*IE_ERunning - MainFlow/Main - *#18*
IE_ERunning - measArray/FirstScan - #88

IE_I 1 INT test
Constant No
External Access: Read/Write
IE_I - calcBreath/Main - #11, #11
IE_I - MainFlow/Main - #17, #9
*IE_I - measArray/FirstScan - #79, *#80, *#82*

IE_IRunning 1 INT test
Constant No
External Access: Read/Write
*IE_IRunning - MainFlow/Main - *#17*
IE_IRunning - measArray/FirstScan - #82

Insp 1 BOOL test

Insp (Continued)
Constant No
External Access: Read/Write
*Insp - ISR_ValveCtrl/Main - #26, #6, #6, *#114, *#12, *#16, *#71, *#9*
*Insp - MainFlow/Main - *#66*

inspAirFlowCtrl REAL[650] test
650 length array with flow values for insp air valve, moved from nextInspAirFlowCtrl[] by ISR_ValveCtrl()
Constant No
External Access: Read/Write
inspAirFlowCtrl - ISR_ValveCtrl/Main - #63
*inspAirFlowCtrl[0] - initVent/Main - *#6*
*inspAirFlowCtrl[0] - ISR_ValveCtrl/Main - *#87*

inspAirFlowCtrlNew REAL[650] test
Constant No
External Access: Read/Write
*inspAirFlowCtrlNew - calcBreath/Main - *#55, *#63, *#66, *#69*
inspAirFlowCtrlNew[0] - calcBreath/Main - #78

inspCount 0 INT test
between 1-650 to indicate where valveCtrl should read from inspAirFlowCtrl[] and inspO2FlowCtrl[]
Constant No
External Access: Read/Write
*inspCount - ISR_ValveCtrl/Main - #11, #41, #61, #63, #7, #70, *#41, *#97*

inspCountLength 100 INT test
between 1-650 to indicate when valveCtrl should stop reading from inspAirFlowCtrl[] and inspO2FlowCtrl[], copied from nextInspCountLength by ISR_ValveCtrl()

Constant No
External Access: Read/Write
*inspCountLength - initVent/Main - #11, *#8*
*inspCountLength - ISR_ValveCtrl/Main - #11, #7, #70, *#88*

inspCountLengthNew 100.0 REAL test
Constant No
External Access: Read/Write
*inspCountLengthNew - calcBreath/Main - #16, #59, #79, *#14*

inspCounts 100.00001 REAL test
Constant No
External Access: Read/Write
*inspCounts - calcBreath/Main - #14, *#13*

InspiratoryRampTimeInMS 200.00002 REAL test
Constant No
External Access: Read/Write
*InspiratoryRampTimeInMS - calcBreath/Main - #22, #27, #43, *#21*

inspO2FlowCtrl REAL[650] test
650 length array with flow values for insp O2 valve, moved from nextInspO2FlowCtrl[] by ISR_ValveCtrl()
Constant No
External Access: Read/Write
inspO2FlowCtrl - ISR_ValveCtrl/Main - #61
*inspO2FlowCtrl[0] - initVent/Main - *#5*
*inspO2FlowCtrl[0] - ISR_ValveCtrl/Main - *#86*

inspO2FlowCtrlNew REAL[650] test
Constant No
External Access: Read/Write
*inspO2FlowCtrlNew - calcBreath/Main - *#54, *#62, *#65, *#68*
inspO2FlowCtrlNew[0] - calcBreath/Main - #77

inspPauseCalc 40.000004 REAL test
Constant No
External Access: Read/Write
*inspPauseCalc - calcBreath/Main - #51, *#50*

inspPauseNrOfAdjustments 40.0 REAL test
Constant No
External Access: Read/Write
*inspPauseNrOfAdjustments - calcBreath/Main - #67, *#51*

inspPauseStartI 0 INT test
Constant No
External Access: Read/Write
*inspPauseStartI - measArray/FirstScan - *#35*
*inspPauseStartI - measCalc/Main - #120, #130, #94, *#29*

inspPauseTime 10 INT test
Constant No
External Access: Read/Write
inspPauseTime - calcBreath/Main - #8
inspPauseTime - MainFlow/Main - #19, #9
*inspPauseTime - measArray/FirstScan - #91, *#92, *#94*

inspPauseTimeInMS 400.00003 REAL test
Constant No
External Access: Read/Write
*inspPauseTimeInMS - calcBreath/Main - #22, #50, *#20*
inspPauseTimeInMS - measCalc/Main - #95

inspPauseTimePercent 0.1 REAL test
Constant No
External Access: Read/Write
*inspPauseTimePercent - calcBreath/Main - #20, *#8*

inspPauseTimeRunning 10 INT test
Constant No
External Access: Read/Write
*inspPauseTimeRunning - MainFlow/Main - *#19*
inspPauseTimeRunning - measArray/FirstScan - #94

inspRampTime 5 INT test
Constant No
External Access: Read/Write
inspRampTime - calcBreath/Main - #9
inspRampTime - MainFlow/Main - #20, #9
*inspRampTime - measArray/FirstScan - #97, *#100, *#98*

inspRampTimePercent 0.05 REAL test
Constant No
External Access: Read/Write
*inspRampTimePercent - calcBreath/Main - #21, *#9*

inspRampTimeRunning 5 INT test
Constant No
External Access: Read/Write
*inspRampTimeRunning - MainFlow/Main - *#20*
inspRampTimeRunning - measArray/FirstScan - #100

inspStartI 0 INT test
Constant No
External Access: Read/Write
*inspStartI - measArray/FirstScan - *#34*
*inspStartI - measCalc/Main - #107, #139, #14, #23, #37, #49, #54, #64, #68, #78, #86, *#19, *#50, *#6*

inspTimeinMS 1000.00006 REAL test
Constant No
External Access: Read/Write
*inspTimeinMS - calcBreath/Main - #13, #15, #22, *#11*

last_state 3 DINT test
Constant No
External Access: Read/Write
*last_state - GUI/Handle_GUI - #50, *#186*

Local:1:O AB:Embedded_DiscreteIO1:O:0 test
Constant No
External Access: Read/Write
Local:1:O.Data.0 1 BOOL
*expOpen - ISR_ValveCtrl/Main - *#100, *#54, *#60*

Local:2:I AB:Embedded_AnalogIO1:I:0 test
Constant No
External Access: Read/Write
Local:2:I.Ch0Data 2#0000_0011_1101_0101 INT
AnalogInputFlowO2 - measArray/measArrays - #111, #36
Local:2:I.Ch1Data 2#0000_0011_1101_0100 INT
AnalogInputFlowAir - measArray/measArrays - #109, #34
Local:2:I.Ch2Data 2#0000_1111_0110_0111 INT
AnalogInputPressure - ISR_ValveCtrl/Main - #1
AnalogInputPressure - measArray/measArrays - #110, #38
Local:2:I.Ch3Data 2#0000_0100_0010_0011 INT
AnalogInputFlow - measArray/measArrays - #32

Local:2:O AB:Embedded_AnalogIO1:O:0 test
Constant No
External Access: Read/Write
Local:2:O.Ch0Data 0 INT
*AnalogOutFlowO2 - ISR_ValveCtrl/Main - *#52, *#62, *#98*
Local:2:O.Ch1Data 0 INT
*AnalogOutFlowAir - ISR_ValveCtrl/Main - *#53, *#64, *#99*

maxFlowAllowed 100 INT test
Constant No
External Access: Read/Write
maxFlowAllowed - calcBreath/Main - #31
*maxFlowAllowed - measArray/FirstScan - *#58*

Mean50 AOI_Mean50 test
Constant No
External Access: Read/Write
*Mean50 - measCalc/Main - *#206, *#208, *#210, *#212, *#214, *#216, *#218, *#220, *#222, *#224*
Mean50.EnableIn 0 BOOL
Enable Input - System Defined Parameter
Mean50.EnableOut 0 BOOL
Enable Output - System Defined Parameter
Mean50.Out_Mean 2.059567 REAL
Mean50.Out_Mean - measCalc/Main - #207, #209, #211, #213, #215, #217, #219, #221, #223, #225

message 'Data Ready' STRING test
Constant No
External Access: Read/Write
*message - calcBreath/Main - *#32*
*message - GUI/Handle_GUI - *#184, *#196*
*message - initVent/Main - *#16*

Messages_to_user STRING[10] test
Constant No
External Access: Read/Write

Messages_to_user (Continued)
Messages_to_user[0] - GUI/Handle_GUI - #184
Messages_to_user[1] - initVent/Main - #16
Messages_to_user[2] - calcBreath/Main - #32

Mode 'VC' STRING test
Constant No
External Access: Read/Write
*Mode - measArray/FirstScan - *#33*
Mode - measCalc/Main - #28

MV 48025.97 REAL test
minute ventilation in l/min
Constant No
External Access: Read/Write
*MV - measCalc/Main - #169, *#113*

MVArr REAL[50] test
50 length array with minute ventilation in l/min, from measCalc()
Constant No
External Access: Read/Write
*MVArr - measArray/FirstScan - *#48*
*MVArr - measCalc/Main - #155, *#155, *#218*
*MVArr[0] - measCalc/Main - *#169*

NewExpOpenCloseAvail 0 BOOL test
signal to valveCtrl to include new expOpenCloseCtrl[]to overwrite when expiration ended, set by calcBreath()
Constant No
External Access: Read/Write
*NewExpOpenCloseAvail - calcBreath/Main - *#87*
NewExpOpenCloseAvail - ISR_ValveCtrl/Main - #215,#225*

newInspFlowAvil 0 BOOL test
Constant No
External Access: Read/Write
*newInspFlowAvil - calcBreath/Main - *#86*
*newInspFlowAvil - ISR_ValveCtrl/Main - #74, *#89*

NewSettings 0 BOOL test
Constant No
External Access: Read/Write
*NewSettings - calcBreath/Main - *#85*
*NewSettings - MainFlow/Main - #42, *#13*
*NewSettings - measArray/FirstScan - *#16*

NewTimeRate_p5 10000 DINT test
Constant No
External Access: Read/Write
*NewTimeRate_p5 - measArray/measArrays - #22, *#19*

NewTimeRate_p8 10000 DINT test
Constant No
External Access: Read/Write
*NewTimeRate_p8 - measArray/measArrays - #22, *#20*

nextExpCountLength 300 INT test
between 1-1000 to indicate when valveCtrl should stop reading from expOpenCloseCtrl[], set by calcBreath()
Constant No
External Access: Read/Write
*nextExpCountLength - calcBreath/Main - *#83*
nextExpCountLength - initVent/Main - #9
nextExpCountLength - ISR_ValveCtrl/Main - #224

nextExpOpenCloseCtrl INT[1000] test

nextExpOpenCloseCtrl (Continued)
1000 length array with open (1) or close (0) for exp valve, set by calcBreath()
Constant No
External Access: Read/Write
*nextExpOpenCloseCtrl[0] - calcBreath/Main - *#82*
nextExpOpenCloseCtrl[0] - initVent/Main - #7
nextExpOpenCloseCtrl[0] - ISR_ValveCtrl/Main - #223

nextInspAirFlowCtrl REAL[650] test
650 length array with flow values for insp air valve, set by calcBreath()
Constant No
External Access: Read/Write
*nextInspAirFlowCtrl[0] - calcBreath/Main - *#78*
nextInspAirFlowCtrl[0] - initVent/Main - #6
nextInspAirFlowCtrl[0] - ISR_ValveCtrl/Main - #87

nextInspCountLength 100 INT test
between 1-650 to indicate when valveCtrl should stop reading from inspAirFlowCtrl[] and inspO2FlowCtrl[], set by calcBreath()
Constant No
External Access: Read/Write
*nextInspCountLength - calcBreath/Main - *#79*
nextInspCountLength - initVent/Main - #8
nextInspCountLength - ISR_ValveCtrl/Main - #88

nextInspO2FlowCtrl REAL[650] test
650 length array with flow values for insp O2 valve, set by calcBreath()
Constant No
External Access: Read/Write
*nextInspO2FlowCtrl[0] - calcBreath/Main - *#77*
nextInspO2FlowCtrl[0] - initVent/Main - #5
nextInspO2FlowCtrl[0] - ISR_ValveCtrl/Main - #86

O2ConstantFlow 3.6455727 REAL test
Constant No
External Access: Read/Write
*O2ConstantFlow - calcBreath/Main - #45, #65, *#40*

O2Down CAM_PROFILE[1010] test
Constant No
External Access: Read/Write
O2Down - ValveProfile/Main - 0(MAPC)

O2mADer 0.0 REAL test
Constant No
External Access: Read/Write
*O2mADer - ISR_ValveCtrl/Main - *#61*

O2mAOut 0.0 REAL test
Constant No
External Access: Read/Write
*O2mAOut - ISR_ValveCtrl/Main - #62, *#61*

O2mASlope 416748.25 REAL test
Constant No
External Access: Read/Write
*O2mASlope - ISR_ValveCtrl/Main - *#61*

O2RampFlowIncreasePerAdjust 0.18227863 REAL test
Constant No
External Access: Read/Write
*O2RampFlowIncreasePerAdjust - calcBreath/Main - #62, *#45*

O2Up CAM_PROFILE[1010] test
Constant No

O2Up (Continued)
External Access: Read/Write
O2Up - ISR_ValveCtrl/Main - #61
O2Up - ValveProfile/Main - 0(MAPC)

PEEP 4402.0 REAL test
positive end expiratory pressure in cmH2O
Constant No
External Access: Read/Write
*PEEP - measCalc/Main - #127, #131, #164, *#68*

PEEPArr REAL[50] test
50 length array with positive end expiratory pressure in cmH2O, from measCalc()
Constant No
External Access: Read/Write
*PEEPArr - measArray/FirstScan - *#43*
*PEEPArr - measCalc/Main - #150, *#150, *#208*
*PEEPArr[0] - measCalc/Main - *#164*

Ppeak 10534.0 REAL test
Ppeak
Constant No
External Access: Read/Write
*Ppeak - measCalc/Main - #121, #127, #163, *#79*

PpeakArr REAL[50] test
50 length array with Ppeak from measCalc(), used for minAvg
Constant No
External Access: Read/Write
*PpeakArr - measArray/FirstScan - *#42*
*PpeakArr - measCalc/Main - #149, *#149, *#206*
*PpeakArr[0] - measCalc/Main - *#163*

PpeakMaxValue AOI_Max test
Constant No
External Access: Read/Write
*PpeakMaxValue - measCalc/Main - *#78*
PpeakMaxValue.EnableIn 0 BOOL
Enable Input - System Defined Parameter
PpeakMaxValue.EnableOut 0 BOOL
Enable Output - System Defined Parameter
PpeakMaxValue.Out_Max 10534.0 REAL
PpeakMaxValue.Out_Max - measCalc/Main - #79

Pplat 9761.556 REAL test
plateau insp pressure in cmH2O
Constant No
External Access: Read/Write
*Pplat - measCalc/Main - #121, #131, #166, *#100, *#102*

PplatArr REAL[50] test
50 length array with plateau insp pressure in cmH2O, from measCalc()
Constant No
External Access: Read/Write
*PplatArr - measArray/FirstScan - *#45*
*PplatArr - measCalc/Main - #152, *#152, *#212*
*PplatArr[0] - measCalc/Main - *#166*

PplatBreathFrac 0.2 REAL test
Constant No
External Access: Read/Write
*PplatBreathFrac - measArray/FirstScan - *#40*
PplatBreathFrac - measCalc/Main - #95

PplatMean AOI_Mean2D test
Constant No
External Access: Read/Write
*PplatMean - measCalc/Main - *#99*
PplatMean.EnableIn 0 BOOL
Enable Input - System Defined Parameter
PplatMean.EnableOut 0 BOOL
Enable Output - System Defined Parameter
PplatMean.Out_Mean 9761.556 REAL
PplatMean.Out_Mean - measCalc/Main - #100

PressArr REAL[1000,2] test
1000 length 2D FIFO array with pressure values and status bits indicating if values have been analysed (1) or not (0) by measCalc()
Constant No
External Access: Read/Write
PressArr - ISR_ValveCtrl/Main - #137, #144, #152, #165, #172, #181, #192, #200
*PressArr - MainFlow/Main - *#58*
*PressArr - measArray/FirstScan - *#9*
*PressArr - measArray/measArrays - *#38, *#39*

PressArrMem REAL[1000,2] test
Constant No
External Access: Read/Write
*PressArrMem - ISR_ValveCtrl/Main - *#144, *#165, *#172, *#181, *#200*
*PressArrMem - MainFlow/Main - *#61*
*PressArrMem - measCalc/Main - #68, *#230, *#78, *#99*
*PressArrMem[0,0] - ISR_ValveCtrl/Main - *#137, *#152, *#192*

PressHtresholdUser1 35.0 REAL ISR_ValveCtrl
Constant No
External Access: Read/Write
PressHtresholdUser1 - ISR_ValveCtrl/Main - #43

Pressure -0.30625 REAL test
Constant No
External Access: Read/Write
*Pressure - ISR_ValveCtrl/Main - #43, *#1*

RampNrCalc 20.000002 REAL test
Constant No
External Access: Read/Write
*RampNrCalc - calcBreath/Main - #44, *#43*

RampNrOfAdjustments 20.0 REAL test
Constant No
External Access: Read/Write
*RampNrOfAdjustments - calcBreath/Main - #45, #46, #60, #64, #64, #67, #67, *#44*

RampTime REAL[650] test
Constant No
External Access: Read/Write
*RampTime - calcBreath/Main - *#61*

Raw 2.440968 REAL test
airway resistance in cmH2O/l/s
Constant No
External Access: Read/Write
*Raw - measCalc/Main - #172, *#121, *#123*

RawArr REAL[50] test
50 length array with airway resistance in cmH2O/l/s
Constant No
External Access: Read/Write
*RawArr - measArray/FirstScan - *#51*

RawArr (Continued)
*RawArr - measCalc/Main - #158, *#158, *#224*
*RawArr[0] - measCalc/Main - *#172*

Round AOI_Round test
Constant No
External Access: Read/Write
*Round - measCalc/Main - *#96*

Round.EnableIn 0 BOOL
Enable Input - System Defined Parameter

Round.EnableOut 0 BOOL
Enable Output - System Defined Parameter

RR 15.0 REAL test
respiratory rate in 1/min
Constant No
External Access: Read/Write
*RR - measCalc/Main - #113, #167, *#65*

RRArr REAL[50] test
50 length array with respiratory rate in 1/min, from measCalc()
Constant No
External Access: Read/Write
*RRArr - measArray/FirstScan - *#46*
*RRArr - measCalc/Main - #153, *#153, *#214*
*RRArr[0] - measCalc/Main - *#167*

RRate 15 INT test
Constant No
External Access: Read/Write
RRate - calcBreath/Main - #10
*RRate - GUI/Handle_GUI - #110, #111, #157, #158, *#110, *#112, *#157, *#159*
RRate - MainFlow/Main - #16, #9
*RRate - measArray/FirstScan - #73, *#74, *#76*

RRateRunning 15 INT test
Constant No
External Access: Read/Write
*RRateRunning - MainFlow/Main - *#16*
RRateRunning - measArray/FirstScan - #76

RunModes STRING[4] test
Constant No
External Access: Read/Write
RunModes[0] - measArray/FirstScan - #33
RunModes[0] - measCalc/Main - #28

SampelRateChangeAllowed 0 BOOL test
Constant No
External Access: Read/Write
*SampelRateChangeAllowed - measArray/FirstScan - *#4*
*SampelRateChangeAllowed - measArray/measArrays - *#23*

Samplerate 100 DINT test
The rate in Hz that samples is collected
Constant No
External Access: Read/Write
*Samplerate - measArray/FirstScan - *#5*
*Samplerate - measArray/measArrays - #12, #15, #18, #6, #8, *#13, *#9*
Samplerate - measCalc/Main - #108, #64, #95

Samplerate_Old 0 DINT test
Constant No
External Access: Read/Write

Samplerate_Old (Continued)
*Samplerate_Old - measArray/FirstScan - *#6*
*Samplerate_Old - measArray/measArrays - #6, *#18, *#24*

SetupDone 1 BOOL test
Constant No
External Access: Read/Write
SetupDone - MainFlow/Main - #29
*SetupDone - measArray/FirstScan - *#15*
*SetupDone - Setup/Main - *#10*

SetupState 0 DINT test
Constant No
External Access: Read/Write
*SetupState - MainFlow/Main - #35, *#31, *#36*
*SetupState - measArray/FirstScan - *#23*
*SetupState - Setup/Main - #1, *#11, *#4, *#7*

Start_Stop 0 BOOL test
Constant No
External Access: Read/Write
*Start_Stop - GUI/Handle_GUI - #16, *#11, *#6*

StartPoint 186 INT test
Constant No
External Access: Read/Write
*StartPoint - measCalc/Main - #99, *#97*

StartPointMean 8.0 REAL test
Constant No
External Access: Read/Write
*StartPointMean - measCalc/Main - #97, *#96*

StartPointValue 8.2 REAL test
Constant No
External Access: Read/Write
*StartPointValue - measCalc/Main - #96, *#95*

StartSendPosition -30 DINT test
Constant No
External Access: Read/Write
*StartSendPosition - ISR_ValveCtrl/Main - #131, #133, #134, #141, #148, #188, #189, #196, *#129, *#187*

StartVent 0 BOOL test
1: start ventilating patient, 0: do nothing
Constant No
External Access: Read/Write
*StartVent - GUI/Handle_GUI - *#19*
*StartVent - initVent/Main - *#14, *#25*
StartVent - MainFlow/Main - #78
*StartVent - measArray/FirstScan - *#17*

state 3 DINT test
Constant No
External Access: Read/Write
*state - GUI/Handle_GUI - #144, #186, #53, #54, #59, #97, *#50, *#54, *#55*

state_selected 0 BOOL test
Constant No
External Access: Read/Write
*state_selected - GUI/Handle_GUI - #138, #49, #91, *#187, *#51*

state0 0 BOOL test
Constant No

state0 (Continued)
External Access: Read/Write
*state0 - GUI/Handle_GUI - *#188, *#60, *#64, *#68, *#72*

state1 0 BOOL test
Constant No
External Access: Read/Write
*state1 - GUI/Handle_GUI - *#189, *#61, *#65, *#69, *#73*

state2 0 BOOL test
Constant No
External Access: Read/Write
*state2 - GUI/Handle_GUI - *#190, *#62, *#66, *#70, *#74*

state3 0 BOOL test
Constant No
External Access: Read/Write
*state3 - GUI/Handle_GUI - *#191, *#63, *#67, *#71, *#75*

StateCalcBreath 0 DINT test
Constant No
External Access: Read/Write
*StateCalcBreath - calcBreath/Main - #5, *#88*
*StateCalcBreath - MainFlow/Main - #44, *#45*
*StateCalcBreath - measArray/FirstScan - *#56*

StateInitVent 0 DINT test
Constant No
External Access: Read/Write
*StateInitVent - initVent/Main - #1, *#12, *#26*
*StateInitVent - MainFlow/Main - #80, *#81, *#87*

StateMain 0 DINT test
Constant No
External Access: Read/Write
*StateMain - MainFlow/Main - #2, #26, *#3, *#30, *#48, *#69, *#74, *#86*
*StateMain - measArray/FirstScan - *#14*

StateValveCtrl 0 DINT test
Constant No
External Access: Read/Write
*StateValveCtrl - ISR_ValveCtrl/Main - #113, #222, #23, #39, #50, #58, #68, #84, #94, *#103, *#109, *#216, *#218, *#226, *#27, *#35, *#44, *#46, *#55, *#65, *#75, *#77, *#80, *#90*

StopVent 0 BOOL test
1: stop ventilating patient after finished breath, shut down not necessary ISRs
Constant No
External Access: Read/Write
*StopVent - GUI/Handle_GUI - *#25*
*StopVent - MainFlow/Main - #52, *#65*
*StopVent - measArray/FirstScan - *#18*

Time 4800 DINT test
Constant No
External Access: Read/Write
*Time - GUI/Handle_GUI - #196, *#195*

Timer 0 DINT test
Constant No
External Access: Read/Write
*Timer - GUI/Handle_GUI - #195, #32, #33, *#141, *#32, *#36, *#45, *#94*

TimeRate 10000 DINT test
Constant No

TimeRate (Continued)
External Access: Read/Write
*TimeRate - measArray/measArrays - #16, #17, #22, #22, *#15*
TimeRate - measCalc/Main - #30

TimerDone 0 BOOL test
Constant No
External Access: Read/Write
*TimerDone - GUI/Handle_GUI - #183, *#192, *#197, *#35*

tipause 250 INT test
Constant No
External Access: Read/Write
tipause - measCalc/Main - #30

totalCount 400.00003 REAL test
Constant No
External Access: Read/Write
*totalCount - calcBreath/Main - #16, *#12*

totTime 0 INT test
Constant No
External Access: Read/Write
*totTime - measArray/FirstScan - *#38*
*totTime - measCalc/Main - #195, #197, *#195*

valveUpdInterval 10.0 REAL test
Constant No
External Access: Read/Write
valveUpdInterval - calcBreath/Main - #12, #13, #43, #48, #50, #61
*valveUpdInterval - measArray/FirstScan - *#57*

VentOn 0 BOOL test
1: keep ventilatin patient, checked by ISR_ValveCtrl() to stop after breath and measCalc()
Constant No
External Access: Read/Write
VentOn - GUI/Handle_GUI - #18, #24
*VentOn - initVent/Main - *#15, *#24*
VentOn - ISR_ValveCtrl/Main - #21
*VentOn - MainFlow/Main - *#55*
*VentOn - measArray/FirstScan - *#19*

Vt 600 INT test
200-600 ml
Constant No
External Access: Read/Write
Vt - calcBreath/Main - #27
*Vt - GUI/Handle_GUI - #104, #105, #151, #152, *#104, *#106, *#151, *#153*
Vt - MainFlow/Main - #15, #9
*Vt - measArray/FirstScan - #67, *#68, *#70*

Vti 3201.7312 REAL test
insp tidal volume in ml
Constant No
External Access: Read/Write
*Vti - measCalc/Main - #108, #113, #127, #131, #168, *#108*

VtiArr REAL[50] test
50 length array with insp tidal volume in ml, from measCalc()
Constant No
External Access: Read/Write
*VtiArr - measArray/FirstScan - *#47*
*VtiArr - measCalc/Main - #154, *#154, *#216*
*VtiArr[0] - measCalc/Main - *#168*

VtRunning	600	INT	test
Constant	No		
External Access:	Read/Write		
*VtRunning - MainFlow/Main - *#15*			
VtRunning - measArray/FirstScan - #70			

General

Configuration

Type:	Periodic	Watchdog:	300.000 ms
Period:	200.000 ms	Disable automatic output processing to reduce task overhead:	No
Priority:	10	Inhibit task:	No

Program Schedule

Scheduled

GUI ValveProfile

Unscheduled

Monitor

Scan Times(Elapsed Time)

Max:	0.486000 ms	Last:	0.299000 ms

Interval Times(Elapsed Times Between Triggers)

Max:	200.207993 ms	Min:	199.792007 ms

Task overlap count: 0

General

Configuration

Main:	Handle_GUI	Inhibit program:	No
Fault:	None	Synchronize redundancy data after execution:	Yes

Monitor

Scan Times(Execution Time)

Max:	67 us	Last:	36 us

```
// Start and stop of ventilator

HMI_ScreenNo :=0;

If HMIStart then
    Start_Stop :=1;
    HMI_ScreenNo :=2;
end_if;

If HMIStop then
   Start_Stop :=0;
   HMI_ScreenNo :=1;
End_if;


if Start_Stop then                                  // If momantary push botton from GUI is on

    if not VentOn then;                             // If VentOn from ISR_ValveCtrl is NOT on
        StartVent := 1;                             // Set start signal to MainFlow sequence
    end_if;


else                                                // If stop from GUI


    if VentOn then;                                 // If Ventilator is running in ISR_ValveCtrl
        StopVent :=1;                               // Set stop signal to MainFlow sequence
    end_if;
end_if;

// Control of the F1 puch botton - Shifts between the 4 input just above the Fx bottons
if data_changed then;                               // start timer if data is changed
    Timer := Timer+1;
    if Timer >= 25 then;


        TimerDone := 1;                             // Set timer done
        Timer := 0;
    end_if;
end_if;

IF F1_pushed_new AND NOT F1_pushed_old Then
    F1_pushed_old := 1;
    IF F1_pushed_new THEN
        IF data_changed THEN                        // reset timer if data if changed repeatedly
            data_changed := 0;
            Timer := 0;                             // reset timer

        END_IF;
        data_changed :=1;                           // signal the timer to start
        IF NOT(state_selected) THEN
            state := last_state;
            state_selected := 1;
        ELSE
            IF state <3 THEN
            state := (state+1);
            ELSE state := 0;
        end_if;

        END_IF;
        CASE state OF
```

```
            0:  state0 := 1;
                state1 := 0;
                state2 := 0;
                state3 := 0;
            1:  state0 := 0;
                state1 := 1;
                state2 := 0;
                state3 := 0;
            2:  state0 := 0;
                state1 := 0;
                state2 := 1;
                state3 := 0;
            3:  state0 := 0;
                state1 := 0;
                state2 := 0;
                state3 := 1;
        ELSE;
        END_CASE;
    END_If;
END_IF;


if not F1_pushed_new AND F1_pushed_old then;
    F1_pushed_old :=0;
end_if;


// Control of the F2 botton - Add value to selected variable

IF F2_pushed_new  AND NOT F2_pushed_old Then
    F2_pushed_old := 1;
    IF (F2_pushed_new AND state_selected) THEN          // increment value if state is selected
        IF data_changed THEN                            // reset timer if data if changed repeatedly
            data_changed :=0;
            Timer := 0;                                 // reset timer
        END_IF;
        data_changed := 1;                              // signal the timer to start
        CASE state OF
            0:  FiO2    := FiO2 + 5;                // increment value in case the O2 is selected
                if FiO2 >= 100 then;
                    FiO2 := 100;
                else;

                end_if;
            1:  Vt  := Vt + 50;                 // increment value in case the Vt is selected
                if Vt >= 600 then;
                    Vt := 600;
                else;


                end_if;
            2:  RRate   := RRate + 1;               // increment value in case the RRin is selected
                if RRate >= 35 then;
                    RRate := 35;
                else;

                end_if;
            3:  IE_counter := IE_counter + 1;       // increment value in case the I:E is selected
                if IE_counter >= 2  then;
                    IE_counter := 2;
```

```
                else;

                end_if;
                COP(IE[IE_counter], I_E, 82);
        ELSE;
        END_CASE;
    END_IF;
END_IF;

if not F2_pushed_new AND F2_pushed_old then;
    F2_pushed_old :=0;
end_if;


// Control of the F3 botton - Sub value to selected variable

IF F3_pushed_new AND NOT F3_pushed_old Then
    F3_pushed_old := 1;
    IF (F3_pushed_new AND state_selected) THEN      // decrement value if state is selected
        IF data_changed THEN                        // reset timer if data if changed repeatedly
            data_changed := 0;
            Timer := 0;                             // reset timer
        END_IF;
        data_changed := 1;                          // signal the timer to start
        CASE state OF
            0:  FiO2    := FiO2 - 5;            // decrement value in case the O2 is selected
                if FiO2 <= 0 then;
                    FiO2 := 0;
                else;
                    //FiO2 := 0;
                end_if;
            1:  Vt  := Vt - 50;             // decrement value in case the Vt is selected
                if Vt <= 200 then;
                    Vt := 200;
                else;

                end_if;
            2:  RRate   := RRate - 1;               // decrement value in case the RRin is selected
                if RRate <= 6 then;
                    RRate := 6;
                else;

                end_if;
            3:  IE_counter := IE_counter - 1;           // increment value in case the I:E is selected
                if IE_counter <= 0  then;
                    IE_counter := 0;
                //COP(IE[IE_counter], I_E, 82);
                else;

                end_if;
                COP(IE[IE_counter], I_E, 82);
        ELSE;
        END_CASE;
    END_IF;
END_IF;

if     F3_pushed_new AND F3_pushed_old then;
   not F3_pushed_old :=0;
```

```
end_if;


// If timer is done then deselect all states for new start

IF TimerDone THEN
    COP(Messages_to_user[0], message, 82);
    data_changed := 0;
    last_state := state;
    state_selected := 0;
    state0 := 0;
    state1 := 0;
    state2 := 0;
    state3 := 0;
    TimerDone := 0;
ELSE
    IF data_changed THEN
        Time := Timer*200;
        COP(Time, message, 82);
        TimerDone :=0;
    END_IF;
END_IF;
```

General

Type:	Structured Text	Number of Lines:	Original 203
In Program:	GUI		

General

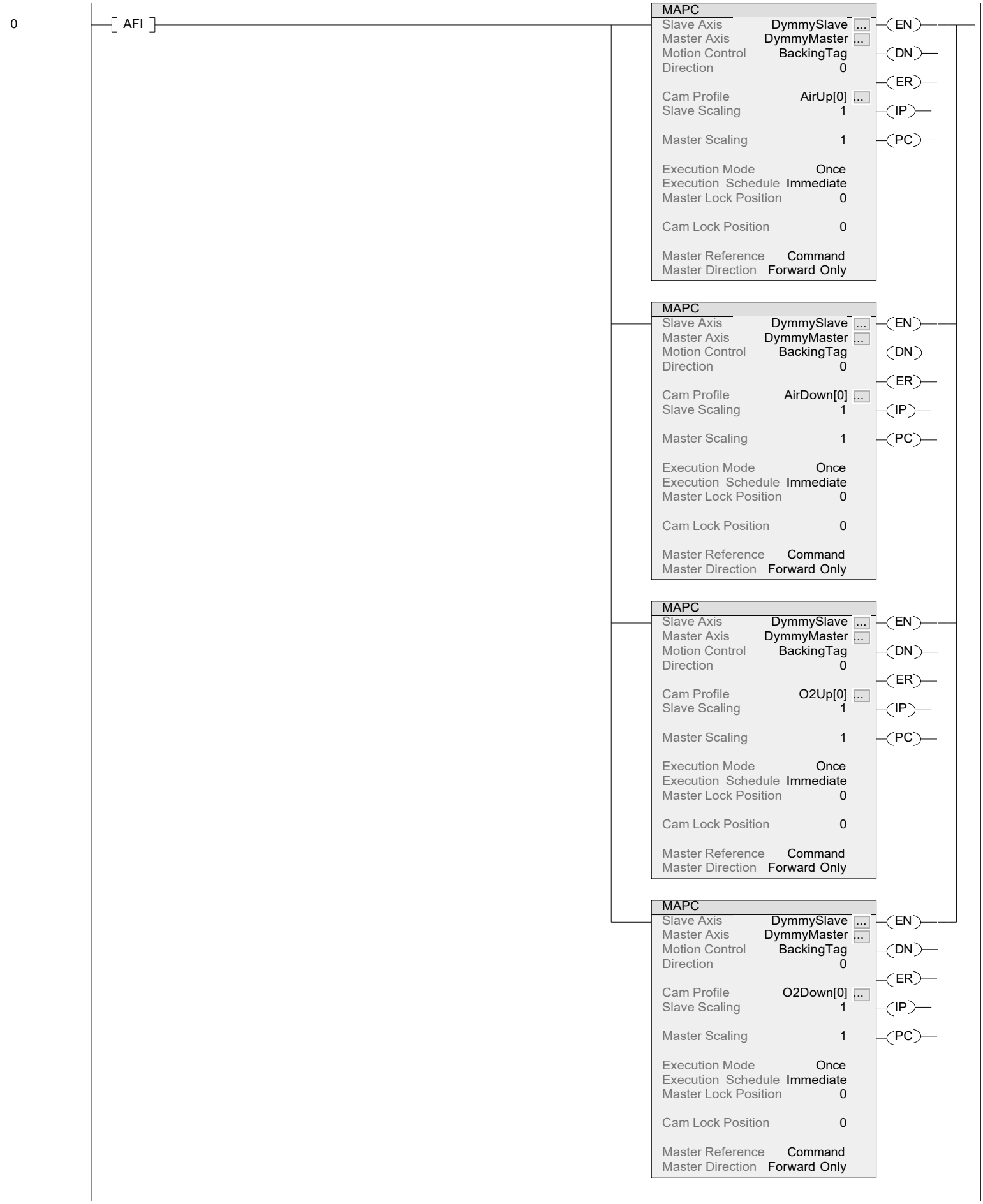
Configuration

Main:	Main	Inhibit program:	No
Fault:	None	Synchronize redundancy data after execution:	Yes

Monitor

Scan Times(Execution Time)

Max:	77 us	Last:	50 us


0
AFI
MAPC
Slave Axis DummySlave
Master Axis DummyMaster
Motion Control BackingTag
Direction 0
Cam Profile AirUp[0]
Slave Scaling 1
Master Scaling 1
Execution Mode Once
Execution Schedule Immediate
Master Lock Position 0
Cam Lock Position 0
Master Reference Command
Master Direction Forward Only
EN
DN
ER
IP
PC
MAPC
Slave Axis DummySlave
Master Axis DummyMaster
Motion Control BackingTag
Direction 0
Cam Profile AirDown[0]
Slave Scaling 1
Master Scaling 1
Execution Mode Once
Execution Schedule Immediate
Master Lock Position 0
Cam Lock Position 0
Master Reference Command
Master Direction Forward Only
EN
DN
ER
IP
PC
MAPC
Slave Axis DummySlave
Master Axis DummyMaster
Motion Control BackingTag
Direction 0
Cam Profile O2Up[0]
Slave Scaling 1
Master Scaling 1
Execution Mode Once
Execution Schedule Immediate
Master Lock Position 0
Cam Lock Position 0
Master Reference Command
Master Direction Forward Only
EN
DN
ER
IP
PC
MAPC
Slave Axis DummySlave
Master Axis DummyMaster
Motion Control BackingTag
Direction 0
Cam Profile O2Down[0]
Slave Scaling 1
Master Scaling 1
Execution Mode Once
Execution Schedule Immediate
Master Lock Position 0
Cam Lock Position 0
Master Reference Command
Master Direction Forward Only
EN
DN
ER
IP
PC

(End)

General

Type:	Ladder Diagram	Number of Rungs:	1
In Program:	ValveProfile		

General

Get raw analog input and control of the breathing - MUST HAVE HIGHEST priority of all peridoc tasks

Configuration

Type:	Periodic	Watchdog:	2000.000 ms
Period:	10.000 ms	Disable automatic output processing to reduce task overhead:	No
Priority:	5	Inhibit task:	No

Program Schedule

Scheduled

measArray ISR_ValveCtrl

Unscheduled

Monitor

Scan Times(Elapsed Time)

Max:	1.412000 ms	Last:	1.258000 ms

Interval Times(Elapsed Times Between Triggers)

Max:	10.943000 ms	Min:	9.867000 ms

Task overlap count: 0

General

Configuration

Main:	Main	Inhibit program:	No
Fault:	None	Synchronize redundancy data after execution:	Yes

Monitor

Scan Times(Execution Time)

Max:	814 us	Last:	420 us

```
// From Analog_values

counter :=0;
SampelRateChangeAllowed := 0;
Samplerate := 100 ;
Samplerate_Old := 0;

AOI_FillArray_Real2D(FillLoop,FlowAir,0,2000);
AOI_FillArray_Real2D(FillLoop,PressArr,0,2000);
//ClearArray_2D(FillLoop,FlowO2);       // Clear all raw data

// Main flow

StateMain := 10;
SetupDone := 0;
NewSettings := 1;
StartVent := 0;
StopVent := 0;
VentOn := 0;

// Setup

SetupState := 0;


// GUI

data_changed :=0;


// measCalc

COP(RunModes[0], Mode, 82); // Set Mode to 'VC'
inspStartI := 0;
inspPauseStartI := 0;
expStartI := 0;
expEndI := 0;
totTime := 0;  //in s
endI := 0;
PplatBreathFrac := 0.2;

AOI_FillArrayReal50(Fill50, PpeakArr, 0, 50);          //Float, 50 length array with Ppeak from measCalc(), used for minAvg
AOI_FillArrayReal50(Fill50, PEEPArr, 0, 50);           //Float, 50 length array with positive end expiratory pressure in cmH2O, from measCalc()
AOI_FillArrayReal50(Fill50, FlowImaxArr, 0, 50);       //Float, 50 length array with max insp flow in l/min, from measCalc()
AOI_FillArrayReal50(Fill50, PplatArr, 0, 50);          //Float, 50 length array with plateau insp pressure in cmH2O, from measCalc()
AOI_FillArrayReal50(Fill50, RRArr, 0, 50);             //Float, 50 length array with respiratory rate in 1/min, from measCalc()
AOI_FillArrayReal50(Fill50, VtiArr, 0, 50);            //Float, 50 length array with insp tidal volume in ml, from measCalc()
AOI_FillArrayReal50(Fill50, MVArr, 0, 50);             //Float, 50 length array with minute ventilation in l/min, from measCalc()
AOI_FillArrayReal50(Fill50, CrsDynArr, 0, 50);         //Float, 50 length array with dynamic respiraotry system compliance in ml/cmH2O, from measCalc()
AOI_FillArrayReal50(Fill50, CrsQStatArr, 0, 50);       //Float, 50 length array with quasistatic respiratory system compliance in ml/cmH2O, from measCalc()
```

```
AOI_FillArrayReal50(Fill50, RawArr, 0, 50);            //Float, 50 length array with airway resistance in cmH2O/l/s, from measCalc()
AOI_FillArrayReal50(Fill50, BreathDurationArr, 0, 50);  //FLOAT, 50 length array with seconds for each breath for calculation of minute avgVentStateArr. Calculated from knowing meas at 100Hz for all breath data

// CalcBreath

StateCalcBreath := 0;       // Reset sequence
valveUpdInterval := 10;     //in ms Frequency 1000/10 = 100Hz
maxFlowAllowed := 100;      //In L/min. Sensor works until 830 ml/s (=49.8 l/min) Note: I will see if we have options to work around this as it will severely limit our options for settings

// See if first time setup is done, the default values else presistent values from controller
if FiO2 = 0 then;
    FiO2 := 4;
else
    FiO2 := FiO2Running;
end_if;

if Vt = 0 then;
    Vt := 350;
else
    Vt := VtRunning;
end_if;

if RRate = 0 then;
    RRate := 15;
else
    RRate := RRateRunning;
end_if;

if IE_I = 0 then;
    IE_I := 1;
else
    IE_I := IE_IRunning;
end_if;

if IE_E = 0 then;
    IE_E := 2;
else
    IE_E := IE_ERunning;
end_if;

if inspPauseTime = 0 then;
    inspPauseTime := 10;
else
    inspPauseTime := inspPauseTimeRunning;
end_if;

if inspRampTime = 0 then;
    inspRampTime := 5;
else
    inspRampTime := inspRampTimeRunning;
end_if;


// Breath detect
_StartPointBerfore := 30;
_StartPosition:= 0;
```

```
_endPosition := 0;
_PointAfterEnd := 30;
```

General

Type:	Structured Text	Number of Lines:	Original 109
In Program:	measArray		

```
// If 1. scan in PLC
if s:fs then;
    jsr(FirstScan,0);
end_if;

// Jump to subroutine
jsr(measArrays,0);
```

General

Type:	Structured Text	Number of Lines:	Original 9
In Program:	measArray		

```
// Enables to change scanrate
// Special on a Rockwell PLC were you can change scan rates on the fly
// Settings can be between 50hz or 100hz - alternativ it can be a fixed setting


if Samplerate <> Samplerate_Old then;

    if Samplerate <= 50 then;
        Samplerate :=50;                        // Set samplerate to 50 Hz if less is choosen
    end_if;


    if Samplerate > 100 then;
        Samplerate := 100;                      // Set samplerate to 100 Hz if higher is choosen
    end_if;
    TimeRate := (1000/Samplerate)*1000;         // Calc from Hz to ms
    SSV(Task,P_Sample_p5,Rate,TimeRate);            // Shift sample rate on data collection
    SSV(Task,P_Sample_p8,Rate,TimeRate);            // Shift sample rate on data collection
    Samplerate_Old := Samplerate;               // Update parameter
    GSV(Task,P_Sample_p5,Rate,NewTimeRate_p5);      // Get the time rate
    GSV(Task,P_Sample_p8,Rate,NewTimeRate_p8);      // Get the time rate

    if TimeRate = NewTimeRate_p5 and TimeRate = NewTimeRate_p8 then;
        SampelRateChangeAllowed := 0;           // Stop change again
        Samplerate_Old := 0;
    end_if;
end_if;



// FIFO fill - load raw analog data into array
if counter >= 0 and counter <= 999 then;                                    // Test counter values is between array
values
    Flow[counter,0] := (AnalogInputFlow*1)/1000;                            // Write Total flow meassure into array 1.
dimension
    Flow[counter,0] := 0;                                                   // Write data read into array 2. dimension
    FlowAir[counter,0] := (AnalogInputFlowAir*75)/1000;                     // Write flowAir meassure into array 1.
dimension
    FlowAir[counter,1] := 0;                                                // Write data read into array 2. dimension
                          (AnalogInputFlowO2*75)/1000;
dimension
    FlowO2[counter,0] :=                                                    // Write flowO2 meassure into array 1.

    FlowO2[counter,1] := 0;                                                 // Write data read into array 2. dimension
dimension
    PressArr[counter,0] := (AnalogInputPressure*6.37322633125)/1000 ;       // Write pressure meassure into array 1.
    PressArr[counter,1] := 0;                                               // Write data read into array 2. dimension


    // ------------------------------------------------------------------------
    // ----------------------   C O D E    F R O M    E D B  ---------------------
    //  All my variables start with _edb.
    // Filtered values are stored in arrays:
    //   _edbFlowArrFiltered
    //   _edbFlowO2Filtered
    //   _edbPressArrFiltered
    // Note that initial setup of filters could be moved in a first scan
    //   initialization routine (code between lines:
    // ********************  Setup filters  ***************************
    //   ...
    // ******************  End of setup filters  ***********************
    //
    // I'm assuming here that all filter
```

```
// are setup in the same way ( same window size, same type, ... ).
// In this isnt true the following code must be modified.

if counter = 0 then

    // ********************  Setup filters  ***************************
    // The lines in this section never change. Hence they could be moved
    // in a first scan initialization.

        // Set the window size of the filter.
        // The total window size will be: 2 * _edbHalfWindowFilterSize + 1
        // Maximum allowed value is 15
        _edbHalfWindowFilterSize := 5; // To be defined



        // Set the type of the filter.
        // Mean filter (1) is basic a convolution and requires setting the
        //   weights
        _edbTypeOfFilter  := 0; // 0: MEDIAN; 1: MEAN
        _edbParUseWeigths := 0; // Not used with a median filter

        // In case _edbTypeOfFilter = 1 and _edbParUseWeigths := 1
        // weights must be set.
        // ...

        // After having decided the above parameters, let's setup the
        //   filters:
        _edbAoiFilterFlowAir.Par_nWindowLeft   := _edbHalfWindowFilterSize;
        _edbAoiFilterFlowAir.Par_nWindowsRight := _edbHalfWindowFilterSize;
        _edbAoiFilterFlowAir.Par_eType
        _edbAoiFilterFlowAir.Par_bUseWeights   := _edbTypeOfFilter;

        _edbAoiFilterFlowO2.Par_nWindowLeft   := _edbHalfWindowFilterSize;
        _edbAoiFilterFlowO2.Par_nWindowsRight := _edbHalfWindowFilterSize;
        _edbAoiFilterFlowO2.Par_eType
        _edbAoiFilterFlowO2.Par_bUseWeights   := _edbTypeOfFilter;

        _edbAoiFilterPress.Par_nWindowLeft   := _edbHalfWindowFilterSize;
        _edbAoiFilterPress.Par_nWindowsRight := _edbHalfWindowFilterSize;
        _edbAoiFilterPress.Par_eType
        _edbAoiFilterPress.Par_bUseWeights   := _edbTypeOfFilter;
    // *****************  End of setup filters  ************************

    // *****************  Init filters' cycle  ************************
    // The following lines must be execute when an acquisition cycle starts
    //  (they cannot be moved in a first scan initialization routine)
    _edbAoiFilterFlowAir.In_bInitialize := 1;
    _edbAoiFilterFlowO2.In_bInitialize := 1;
    _edbAoiFilterPress.In_bInitialize := 1;
    // **************  End of init filters' cycle  *********************

end_if;

// Calculate filtered values
AOI_FilterSignal( _edbAoiFilterFlowAir, AnalogInputFlowAir, _edbParWeights );
AOI_FilterSignal( _edbAoiFilterPress, AnalogInputPressure, _edbParWeights );
AOI_FilterSignal( _edbAoiFilterFlowO2, AnalogInputFlowO2, _edbParWeights );

// Check if initial buffering has finished so we can start storing values.
```

```
    // First _edbHalfWindowFilterSize elements of the array won't be calculated.
    // Last _edbHalfWindowFilterSize elements of the array won't be calculated as well.
    _edbCanStore := not _edbAoiFilterFlowAir.Out_Buffering and
                    not _edbAoiFilterFlowO2.Out_Buffering  and
                    not _edbAoiFilterPress.Out_Buffering   and
                    counter >= 2 * _edbHalfWindowFilterSize; // This last condition should be true when the previouos conditions are true.
    if _edbCanStore then
        _edbIdx := counter - _edbHalfWindowFilterSize; // Calculated filtered data is delayed by _edbHalfWindowFilterSize acquisitions
        _edbFlowArrFiltered[_edbIdx,0] := _edbAoiFilterFlowAir.Out_rFilteredSignalDelayed;
        _edbFlowArrFiltered[_edbIdx,1] := 0;
        _edbFlowO2Filtered[_edbIdx,0] := _edbAoiFilterPress.Out_rFilteredSignalDelayed;
        _edbFlowO2Filtered[_edbIdx,1] := 0;
        _edbPressArrFiltered[_edbIdx,0] := _edbAoiFilterFlowO2.Out_rFilteredSignalDelayed;
        _edbPressArrFiltered[_edbIdx,1] := 0;
    end_if;
    // --------------  E N D   O F   C O D E   F R O M   E D B  ---------------
    // ------------------------------------------------------------------------


    counter := counter +1;                              // next count
end_if;


if counter >= 999 then;                                 // If count is at end of array
    counter := 0;                                       // Reset counter
end_if;
```

General

Type:	Structured Text	Number of Lines:	Original 138
In Program:	measArray		

General

Configuration

Main:	Main	Inhibit program:	No
Fault:	None	Synchronize redundancy data after execution:	Yes

Monitor

Scan Times(Execution Time)

Max:	797 us	Last:	37 us

```
Pressure := 100.0/16000.0 * (AnalogInputPressure - 4000.0);
//PressHtreshold := ((PressHtresholdUser1/100.0)*16000.0)+4000.0;


// Control that either inspiratory or expiratory is set
if (Insp and Exp) or (not Insp and not Exp) then;
    if inspCount < inspCountLength and expCount < expCountLength then;
        Exp := 1;                                           // Select expiatory
        Insp := 0;                                          // Deselect inspiratory
    end_if;
    if inspCount < inspCountLength then;
        Insp := 1;                                          // Select inspiraty
        Exp := 0;                                           // DeSelect expiatory
    end_if;
    if expCount < expCountLength then;
        Insp := 0;                                          // Deselect inspiratory
        Exp := 1;                                           // Select expiatory
    end_if;
end_if;

if VentOn then;

    if StateValveCtrl = 0 then;

        // Control if Inspiratory or expiatory
        if Insp then;
            StateValveCtrl :=10;                                // Inspiratory
            if _StartPosition <> _StartpositionOld then;
                _StartPosition := counter;
                _StartpositionOld := _StartPosition;
            end_if;
        end_if;

        if Exp then
            StateValveCtrl :=100;                               // Expiatory
        end_if;
    End_if;

    if StateValveCtrl = 10 then;

        // Start of the Inspiratory
        inspCount := inspCount +1;                          // Count inspiratory up one in the cycle
        expCount := 0;                                      // Reset expiatory counter
input   if Pressure > PressHtresholdUser1 then;                      // Analog pressure High warning is high from
            StateValveCtrl := 11;                           // Go to high pressure routine
        else
            StateValveCtrl := 20;                           // Go to normal routine
        end_if;
    end_if;

    if StateValveCtrl = 11 then;
        // Close valves - high alarm on the Analog input will disapear when preassure is lower than high alarm
        AnalogOutFlowO2 := 0;                                       // Close O2 valve
        AnalogOutFlowAir := 0;                                      // Close Air Valve
        expOpen :=1;                                        // Output - Open expiatory valve
        StateValveCtrl := 30;                               // Next state
    end_if;

    if StateValveCtrl = 20 then;
```

```
    // Normal routine - set values to open valve
    expOpen := 0;                                           // Output - close the expiatory valve
    MCSV(GetSlaveValue, O2Up, inspO2FlowCtrl[inspCount], O2mAOut, O2mASlope, O2mADer);
    AnalogOutFlowO2 := 1.03*O2mAOut;                                  // Set value to analog output control of O2
    MCSV(GetSlaveValue, AirUp, inspAirFlowCtrl[inspCount], AirmAOut, AirmASlope, AirmADer);
    AnalogOutFlowAir := 1.03*AirmAOut;                              // Set value to analog output control of Air
    StateValveCtrl := 30;                                   // Next state
end_if;

if StateValveCtrl = 30 then;
    // Control if inspiratory has ended
    if inspCount >= inspCountLength then;                   // The Inspiratory has ended
        Insp := 0;                                          // Deselect inspiratory
        Exp := 1;                                           // Select expiatory

        if newInspFlowAvil then;                            // New Inspiratory data is avalible
            StateValveCtrl :=31;                            // Set to new data for next cyclus is avalible
        else
            StateValveCtrl :=0;                             // Start over
        end_if;
    else
        StateValveCtrl :=0;                                 // Start over
    end_if;
end_if;

if StateValveCtrl = 31 then;


    COP(nextInspO2FlowCtrl[0], inspO2FlowCtrl[0], 650);     // Copy new O2 data to next inspiratory
    COP(nextInspAirFlowCtrl[0], inspAirFlowCtrl[0], 650);   // Copy new Air data to next inspiratory
    inspCountLength := nextInspCountLength;                 // Copy new lenght of inspiratory
                    := 0;
    newInspFlowAvil := 0;                                   // Reset new Inspiratory data is avalible
    StateValveCtrl                                          // Start over
end_if;

if StateValveCtrl = 100 then;

    // Start of the Expiatory
    expCount := expCount +1;                                    // Count expiratory up one in the cycle
    inspCount := 0;                                             // Reset inspiratory counter
    AnalogOutFlowO2 := 0;                                               // Close Inspiratory O2 valve
    AnalogOutFlowAir := 0;                                              // Close Inspiratory Air valve
    expOpen := expOpenCloseCtrl[expCount].0;                    // Set state of expiratory valve
    if              expCountLength then;
       expCount >= StateValveCtrl :=101;                        // The Expiratory has ended
        if _endPosition    _endPositionOld then;
                        <>= counter;
            _endPosition := _endPosition;
        end_if;
    else
        StateValveCtrl :=0;                                     // Start over
    end_if;
end_if;

if StateValveCtrl = 101 then;
    Insp :=1;                                                   // Select inspiratory
    Exp := 0;                                                   // DeSelect expiatory

    if flowCalcOn then;
```

```
_StartPointBerfore := _StartPointBerfore;
_StartPosition:= _StartPosition;
_endPosition := _endPosition ;
_PointAfterEnd := _PointAfterEnd;


if (_endPosition -_StartPosition) + _StartPointBerfore +_PointAfterEnd < 999 then;

    if _endPosition - _StartPosition > 0 then

        StartSendPosition := _StartPosition - _StartPointBerfore;

        if StartSendPosition < 0 then

        _CopSource := 999 + StartSendPosition;
        _CopLength := abs(StartSendPosition+1)*2;
        COP(FlowAir[_CopSource,0],FlowArrMem[0,0],_CopLength);
        COP(FlowO2[_CopSource,0],FlowO2Mem[0,0],_CopLength);
        COP(PressArr[_CopSource,0],PressArrMem[0,0],_CopLength);

        _CopSource1 := 0;
        _CopDes1 := _CopLength + 1;
        _CopLength1 := abs(StartSendPosition)+1;
        COP(FlowAir[_CopSource1,0],FlowArrMem[_CopDes1 ,0],_CopLength1);
        COP(FlowO2[_CopSource1,0],FlowO2Mem[_CopDes1 ,0],_CopLength1);
        COP(PressArr[_CopSource1,0],PressArrMem[_CopDes1 ,0],_CopLength1);

    else

        _CopSource3 := StartSendPosition+1;
        _CopLength3 :=_StartPointBerfore*2;
        COP(FlowAir[_CopSource3,0],FlowArrMem[0,0],_CopLength3);
        COP(FlowO2[_CopSource3,0],FlowO2Mem[0,0],_CopLength3);
        COP(PressArr[_CopSource3,0],PressArrMem[0,0],_CopLength3);

    end_if;

    _EndSendPosition := _endPosition + _PointAfterEnd;

    If _EndSendPosition > 999 then

        _CopSource4 := _StartPosition + 1;
        _CopDes4 := _StartPointBerfore;
        _CopLength4 := ( 999 - _StartPosition + 1)*2;
        COP(FlowAir[_CopSource4,0],FlowArrMem[_CopDes4,0],_CopLength4);
        COP(FlowO2[_CopSource4,0],FlowO2Mem[_CopDes4,0],_CopLength4);
        COP(PressArr[_CopSource4,0],PressArrMem[_CopDes4,0],_CopLength4);

        _CopSource5 := 0;
        _CopDes5 := _CopDes4 + 999 - _StartPosition;
        _CopLength5 := (_EndSendPosition - 999) *2;
        COP(FlowAir[_CopSource5,0],FlowArrMem[_CopDes5,0],_CopLength5);
        COP(FlowO2[_CopSource5,0],FlowO2Mem[_CopDes5,0],_CopLength5);
        COP(PressArr[_CopSource5,0],PressArrMem[_CopDes5,0],_CopLength5);

    else

        _CopSource6 := _StartPosition;
```

```
                _CopDes6 := _StartPointBerfore;
                _CopLength6 := (_EndSendPosition - _StartPosition) * 2;
                COP(FlowAir[_CopSource6,0],FlowArrMem[_CopDes6,0],_CopLength6);
                COP(FlowO2[_CopSource6,0],FlowO2Mem[_CopDes6,0],_CopLength6);
                COP(PressArr[_CopSource6,0], PressArrMem[_CopDes6,0],_CopLength6);

            End_if;

            else

                StartSendPosition := _StartPosition - _StartPointBerfore;
                _CopSource7 := StartSendPosition;
                _CopLength7 := (999 - StartSendPosition + 1) *2;
                COP(FlowAir[_CopSource7,0],FlowArrMem[0,0],_CopLength7);
                COP(FlowO2[_CopSource7,0],FlowO2Mem[0,0],_CopLength7);
                COP(PressArr[_CopSource7,0], PressArrMem[0,0],_CopLength7);


                _EndSendPosition := _endPosition + _PointAfterEnd;
                _CopSource8 := 0;
                _CopDes8 := (999 - _StartSendPosition +1);
                _CopLength8 := _EndSendPosition *2;
                COP(FlowAir[_CopSource8,0],FlowArrMem[_CopDes8,0],_CopLength8);
                COP(FlowO2[_CopSource8,0],FlowO2Mem[_CopDes8,0],_CopLength8);
                COP(PressArr[_CopSource8,0], PressArrMem[_CopDes8,0],_CopLength8);

            end_if;
        end_if;

        _endPositionOld :=0;
        _StartpositionOld := 0;
        CalcNewData := 1;

    else

     CalcNewData := 0;

    end_if;


    if NewExpOpenCloseAvail then;                                   // New Expiatory data is avalible
        StateValveCtrl := 102;                                      // Next state
    elseStateValveCtrl := 0;                                        // Start over
    end_if;
end_if;

if StateValveCtrl = 102 then;
    COP(nextExpOpenCloseCtrl[0], expOpenCloseCtrl[0], 1000);        // Copy new close/open data to next expiatory


    expCountLength := nextExpCountLength;                           // Copy new lenght of expiatory
    NewExpOpenCloseAvail := 0;                                      // Reset Expiatory data is avalible
    StateValveCtrl := 0;                                            // Start over
end_if;
end_if;
```

General

Type:	Structured Text	Number of Lines:	Original 229
In Program:	ISR_ValveCtrl		

General

Configuration

Type:	Periodic	Watchdog:	500.000 ms
Period:	10.000 ms	Disable automatic output processing to reduce task overhead:	No
Priority:	8	Inhibit task:	No

Program Schedule

Scheduled

Setup	calcBreath	initVent	BreathDetect
measCalc	MainFlow		

Unscheduled

Monitor

Scan Times(Elapsed Time)

Max:	4.712000 ms	Last:	1.271000 ms

Interval Times(Elapsed Times Between Triggers)

Max:	10.851000 ms	Min:	9.203000 ms

Task overlap count: 0

General

Configuration

Main:	Main	Inhibit program:	No
Fault:	None	Synchronize redundancy data after execution:	Yes

Monitor

Scan Times(Execution Time)

Max:	37 us	Last:	25 us

```
// Init program of the GUI (HMI panel)
// Write initial coder here
```

General

Type:	Structured Text	Number of Lines:	Original 6
In Program:	Setup		

```
// Init program of the ISR_RS232 AI controller
// Write initial coder here
```

General

Type:	Structured Text	Number of Lines:	Original 5
In Program:	Setup		

```
// Init program of the Init varables for the program
// Write initial coder here
```

General

Type:	Structured Text	Number of Lines:	Original 3
In Program:	Setup		

```
case SetupState of

    10: JSR(InitVariables,0);           // Jump to Init Variables
        SetupState := 20;               // Set next state

    20: JSR(InitGUI,0);                 // Jump to Init GUI
        SetupState := 30;               // Set next state

    30: JSR(InitISR_RS232);             // Jump to init ISR_RS232
        SetupDone :=1;                  // Set Setup Done
        SetupState := 99;               // Sequence Done

end_case;
```

General

Type:	Structured Text	Number of Lines:	Original 13
In Program:	Setup		

General

Configuration

Main:	Main	Inhibit program:	No
Fault:	None	Synchronize redundancy data after execution:	Yes

Monitor

Scan Times(Execution Time)

Max:	2606 us	Last:	8 us

```
//Saves o2 and air inspiratory flows in l/min as 100 Hz arrays for ISR_valveCtrl() to
//read and convert to a current securing these defined flows delivered to
//the patient.

if StateCalcBreath = 10 then;

    //Calculate durations
    inspPauseTimePercent := inspPauseTime/100;  //Percentage of whole inspIE_In where there is a pause (in the end)
    inspRampTimePercent := inspRampTime/100;    //Percentage of active inspIE_In (when there is flow)
    BreathDurationInMS := (1/RRate) * 60000;
    inspTimeinMS := IE_I * (BreathDurationInMS/(IE_I+IE_E));
    totalCount := BreathDurationInMS/valveUpdInterval;
    inspCounts := inspTimeinMS/valveUpdInterval;
    AOI_Floor(Floor,inspCounts,inspCountLengthNew);
    expTimeInMS := BreathDurationInMS - inspTimeinMS;
    expCountlengthCalc := totalCount-inspCountLengthNew;
    AOI_Floor(Floor, expCountlengthCalc, expCountLengthNew);

    //Calculate part of inspIE_In where valves are open
    inspPauseTimeInMS := inspPauseTimePercent * BreathDurationInMS;
    InspiratoryRampTimeInMS := inspRampTimePercent * BreathDurationInMS;
    ConstantFlowTimeInMS := inspTimeinMS - inspPauseTimeInMS - InspiratoryRampTimeInMS;


    //Calculate flow in L/min required to reach Vt in active inspIE_In
    //(assuming a linear ramp)
    ConstantInspiratoryFlow := 1/(ConstantFlowTimeInMS+0.5*InspiratoryRampTimeInMS)*Vt*60;

    //This needs to be warned to the user - We will need a handle in HMI to not
    //allow such settings if we are stuck with this limit on max flow
    if ConstantInspiratoryFlow > maxFlowAllowed then;
        COP(Messages_to_user[2], message, 82);
    end_if;

    //Calculate split between O2 and Air
    FractionAir := ((1-FiO2/100)/0.79);
    FractionO2 := 1 - FractionAir;


    //Calculate Ramp samples for O2 and Air
    O2ConstantFlow := FractionO2 * ConstantInspiratoryFlow;         //L/min
    AirConstantFlow := FractionAir * ConstantInspiratoryFlow;       //L/min
    RampNrCalc := InspiratoryRampTimeInMS/valveUpdInterval;
    AOI_Floor(Floor, RampNrCalc,RampNrOfAdjustments);
    O2RampFlowIncreasePerAdjust := O2ConstantFlow/RampNrOfAdjustments;
    AirRampFlowIncreasePerAdjust := AirConstantFlow/RampNrOfAdjustments;

    ConstansFlowNrCalc := ConstantFlowTimeInMS/valveUpdInterval;
    AOI_Floor(Floor, ConstansFlowNrCalc,ConstantFlowNrOfAdjustments );
                    inspPauseTimeInMS/valveUpdInterval;

    inspPauseCalc :=
    AOI_Floor(Floor, inspPauseCalc,inspPauseNrOfAdjustments );
    //Build control arrays
    AOI_FillArrayReal(Fill, inspO2FlowCtrlNew, 0, 650);
    AOI_FillArrayReal(Fill, inspAirFlowCtrlNew, 0, 650);
    AOI_FillArrayInt(Fillint, expOpenCloseCtrlNew, 0, 1000);

    //inspiratory
    for i :=0 to inspCountLengthNew by 1 do;
```

```
        if i <= RampNrOfAdjustments then;
            RampTime[i] := valveUpdInterval*(i-1);
            inspO2FlowCtrlNew[i] := i *  O2RampFlowIncreasePerAdjust;
            inspAirFlowCtrlNew[i] := i *  AirRampFlowIncreasePerAdjust;
        elsif i > RampNrOfAdjustments and i < RampNrOfAdjustments+ConstantFlowNrOfAdjustments then;
            inspO2FlowCtrlNew[i] := O2ConstantFlow;
            inspAirFlowCtrlNew[i] := AirConstantFlow;
        elsif i > (RampNrOfAdjustments+ConstantFlowNrOfAdjustments) and i <= (RampNrOfAdjustments
+ConstantFlowNrOfAdjustments+inspPauseNrOfAdjustments) then;
            inspO2FlowCtrlNew[i] := 0;
            inspAirFlowCtrlNew[i] := 0;
        end_if;
    end_for;

    //expiratory
    AOI_FillArrayInt(Fillint, expOpenCloseCtrlNew, 1, expCountLengthNew);

    // Copy calculated data for Inspiratoty to next breath cycles
    COP(inspO2FlowCtrlNew[0], nextInspO2FlowCtrl[0], 650);
    COP(inspAirFlowCtrlNew[0], nextInspAirFlowCtrl[0], 650);
    nextInspCountLength := inspCountLengthNew;

    // Copy calculated data for Expiratory to next breath cycles
    COP(expOpenCloseCtrlNew[0], nextExpOpenCloseCtrl[0], 1000);
    nextExpCountLength := expCountLengthNew;

    NewSettings := 0;
    newInspFlowAvil := 1;
    NewExpOpenCloseAvail:= 1;
    StateCalcBreath := 0;

end_if;
```

General

Type:	Structured Text	Number of Lines:	Original 92
In Program:	calcBreath		

General

Configuration

Main:	Main	Inhibit program:	No
Fault:	None	Synchronize redundancy data after execution:	Yes

Monitor

Scan Times(Execution Time)

Max:	195 us	Last:	10 us

```
// Init of Valve Control
// Routine only runs once

// Fill BreathDurationArr with -1 only on first breath, for messuring average values
AOI_FillArrayReal50(Fill50,BreathDurationArr, -1, 50);
flowCalcOn := 0;        // No calculation of messurement - set to 1 if messurement ***** TEMP Soulution *****
```

General

Type:	Structured Text	Number of Lines:	Original 8
In Program:	initVent		

```
// Init of Pressure Watch
// Routine only runs once
```

General

Type:	Structured Text	Number of Lines:	Original 2
In Program:	initVent		

```
case StateInitVent of

    10:                                                     // Prepare for transfer of data from
calcBreath routine

        COP(nextInspO2FlowCtrl[0], inspO2FlowCtrl[0], 650);     // Copy calculated data to running data
        COP(nextInspAirFlowCtrl[0], inspAirFlowCtrl[0], 650);   // Copy calculated data to running data
        COP(nextExpOpenCloseCtrl[0], expOpenCloseCtrl[0], 1000); // Copy calculated data to running data
        inspCountLength := nextInspCountLength;             // Copy lenght of inspiratory array
calculated to runnung data
        expCountLength := nextExpCountLength;               // Copy lenght of expiratory array
calculated to runnung data

        if inspCountLength >20 and expCountLength >20 then;  // Check values is correct
            StateInitVent := 20;                            // Set next stage
        else
            StartVent :=0;                                  // Stop ventilation
            VentOn := 0;                                    // Set do not run to ISR_ValveCtrl
            COP(Messages_to_user[1], message, 82);          // Write to user that lenght insp or exp is
not ok
        end_if;


    20:
        JSR(initISR_ValveCtrl);                             // Jump to init for ISR_ValveCtrl - only
runs once
        JSR(Init_ISR_PressWatchID);                         // Jump to init for ISR_PressWatchID - only
runs once

        VentOn := 1;                                        // Set start to ISR_ValveCtrl
        StartVent := 0;                                     // Reset start bit
        StateInitVent := 99;                                // Set sequence finished state

end_case;
```

General

Type:	Structured Text	Number of Lines:	Original 28
In Program:	initVent		

General

Configuration

Main:	Main	Inhibit program:	No
Fault:	None	Synchronize redundancy data after execution:	Yes

Monitor

Scan Times(Execution Time)

Max:	37 us	Last:	7 us

General

Type:	Structured Text	Number of Lines:	Original 1
In Program:	BreathDetect		

General

Configuration

Main:	Main	Inhibit program:	No
Fault:	None	Synchronize redundancy data after execution:	Yes

Monitor

Scan Times(Execution Time)

Max:	36 us	Last:	18 us

```
// from inflow and press alone (may require filtering such as mean/median!)
// Only check indices with FlowArr[0,x]=0, meaning not analysed yet

if CalcNewData then;

    inspStartI := 0;
    expStartI := 0;
    expEndI := 0;

    for _i := 1 to 999 by 1 do

        if FlowArrMem[_i,1] = 0 then;                                   // Check if data is read already

            if inspStartI = 0 then;                                     // If Inspiratory start point
not found

                //Start of insp is at shift from flow of zero to flow of >0.

                if FlowArrMem[_i-1,0] = 0 and FlowArrMem[_i,0] > 0 then;        // If start of climing flow,
were last point was zero and this point has a value
                    inspStartI := _i ;                                  // Set Inspiratory start point
                end_if;
            end_if;

            if inspStartI > 0 and expStartI = 0 then;                   // If Inspiratory start point
found but nor the Expiratory point

                //Start of exp is at shift from to flow of >0 to flow of zero.

                if FlowArrMem[_i-1,0] > 0 and FlowArrMem[_i,0] = 0 then;        // If end of climing flow, and
start of Pplat + Pplat time
                    if Mode = RunModes[0] then;                         // If mode is VC
                        inspPauseStartI := _i;                          // Set inspiratory pause start
point

                        expStartI := _i + ABS(tipause/(TimeRate/1000));         // Set Expiratory start point
                    else
                        expStartI := _i;                                // Set Expiratory start point
                    end_if;
                end_if;
            end_if;

            if inspStartI > 0 and expStartI > 0 and expEndI = 0 then;    // If Inspiratory start point
found and the Expiratory start point, but not Expiratory end point

                //End of exp is at shift from to flow of zero to flow of >0.

                if FlowArrMem[_i-1,0] = 0 and FlowArrMem[_i,0] > 0 then;        // If flow is zero and detection
of new breath
                    expEndI := _i - 1;                                  // Set Expiratory end point
                end_if;
            end_if;
        end_if;
    end_for;

    // Check of iligal calculation that would bring the controller in Major fault
    if inspStartI <1 then;
        inspStartI :=1 ;
    end_if;
```

```
if expStartI <1 then;
    expStartI := inspStartI +1;
end_if;

if expEndI < 1 then;
    expEndI := expStartI +1;
end_if;

// Set all calculations vaerables for breath

// Breathduration
BreathDuration := ((expEndI-inspStartI+1)/Samplerate);
RR    := (60/BreathDuration);

// in area with inflow=0 before insp start --> PEEP
PEEP := PressArrMem[inspStartI-1,0];

// in area with inflow=0 check for increase in press (Exp pause, exp flow=0) --> PEEP+PEEPi
// NOTE as exp pause cannot be detected this is not possible at the moment,
// but should be done here
//PEEP := (PEEP+PEEPi)-PEEP;

// Ppeak = Max pressure during insp
// NOTE: will require filtering, e.g. moving average

AOI_Max(PpeakMaxValue,PressArrMem, 0, inspStartI, expStartI);
Ppeak := PpeakMaxValue.Out_Max;

// Flowi,max = max flow during insp
// NOTE: will require filtering e.g. moving average, should be around same
// sample range as Ppeak
// in VC this is the constant delived level of insp flow

AOI_Max(FlowImaxValue, FlowArrMem, 0,inspStartI, expStartI);
FlowImax := FlowImaxValue.Out_Max;

// Pplat = pressure at period immediately following inFlow=0 where
// pressure equilibrates
// only available if insp pause was performed
// Calculated as mean of last fifth of the pause period

                        then;
                        (inspPauseTimeInMS/1000) * Samplerate*PplatBreathFrac;
if inspPauseStartI > 0
    AOI_Round(Round, StartPointValue, StartPointMean);
    StartPointValue :=
    StartPoint := expStartI-StartPointMean-1;
    EndPoint := expStartI-1;
AOI_Mean2D(PplatMean, PressArrMem, 0, StartPoint, EndPoint);
    Pplat := PplatMean.Out_Mean;
else
    Pplat := 0;
end_if;

//Vti = integrate inFlow(insp start to exp start)



end_for;= inspStartI to expStartI by 1 do
    Vti := Vti + FlowArrMem[_J,0] * (1/(Samplerate*60));
//MV = RR*Vti
```

```
MV := RR*Vti;

//Raw = (Ppeak-Pplat)/Flowi,max    %NOTE: in cmH2O/l/s (different time
//unit than Flowi,max at l/min
//only available if insp pause was performed so that we have Pplat
//assumes FlowIMax is at PIP

if inspPauseStartI > 0 then;
    Raw := (Ppeak-Pplat)/(FlowImax/60);   //FlowIMax/60 to go from l/min to l/s as Raw is in units of l/s
else
    Raw := 0;
end_if;

//Crs_dyn = Vt/(Ppeak-PEEP)
CrsDyn := Vti/(Ppeak-PEEP);

//Crs_qstat = Vt/(Pplat-PEEP)
if inspPauseStartI > 0 then;
    CrsQStat := Vti/(Pplat-PEEP);
else
    CrsQStat := 0;
end_if;


//set second dimension status marker in FlowArr and PressArr to 1 from
//inspStartI to expEndI to indicate that this breath has been analysed
for _n := inspStartI to expEndI+1 by 1 do

    //FlowArrMem[_n,1] := 1;
    //PressArrMem[_n,1] := 1;

end_for;


for _T := 49 to 1 by -1 do

    PpeakArr[_T] := PpeakArr[_T-1];
    PEEPArr[_T] := PEEPArr[_T-1];
    FlowImaxArr[_T] := FlowImaxArr[_T-1];
    PplatArr[_T] := PplatArr[_T-1];
    RRArr[_T] := RRArr[_T-1];
    VtiArr[_T] := VtiArr[_T-1];
    MVArr[_T] := MVArr[_T-1];
    CrsDynArr[_T] := CrsDynArr[_T-1];
    CrsQStatArr[_T] := CrsQStatArr[_T-1];
    RawArr[_T] := RawArr[_T-1];
    BreathDurationArr[_T] := BreathDurationArr[_T-1];

end_for;

                := Ppeak;
    PpeakArr[0] := PEEP;
    PEEPArr[0]  := FlowImax;
                := Pplat;
    RRArr[0]    := RR;
    PplatArr[0] := Vti;
    VtiArr[0] := MV;
    MVArr[0] := CrsDyn;
    CrsDynArr[0] := CrsQStat;
```

```
        CrsQStatArr[0] := CrsQStat;
        RawArr[0] := Raw;
        BreathDurationArr[0] := BreathDuration;


    //in case there is not a minute of measurements available yet, then calculate
    //from the available entries in array
    //This could be signaled by setting all values in array as -1 until they
    //have been used, this is done by initVent
    //I asume here that if an entry is -1 then it is not available

    for _z := 1 to 49 by 1 do
        if BreathDurationArr[_z] = -1 and _z = 1 then;
            //(disp('error, no breath by breath measurements available');
            _z :=50;                                        // Break the for loop
        end_if;
        if _z < 50 then;
            if BreathDurationArr[_z] = -1 then;
                endI := i-1;
                _z :=50;
            end_if;
        end_if;

        if _z < 49 then;
            totTime := totTime+BreathDurationArr[_z];
        end_if;
        if totTime >= 60 then;
            endI := _z;
            _z :=50;
        end_if;
    end_for;


    // Calculate the average from last minute


    AOI_Mean50(Mean50, PpeakArr, 1, endI);          //1 min avg of peak insp pressure in cmH2O
    avgPpeak :=Mean50.Out_Mean;                     // Output of mean
    AOI_Mean50(Mean50, PEEPArr, 1, endI);           //1 min avg of positive end expiratory pressure in cmH2O
    avgPEEP :=Mean50.Out_Mean;                      // Output of mean
    AOI_Mean50(Mean50, FlowImaxArr, 1, endI);       //1 min avg of max insp flow in l/min
    avgFlowImax :=Mean50.Out_Mean;                      // Output of mean
    AOI_Mean50(Mean50, PplatArr, 1, endI);          //1 min avg of plateau insp pressure in cmH2O
    avgPplat :=Mean50.Out_Mean;                     // Output of mean
    AOI_Mean50(Mean50, RRArr, 1, endI);             //1 min avg of respiratory rate in 1/min
    avgRR :=Mean50.Out_Mean;                        // Output of mean
    AOI_Mean50(Mean50, VtiArr, 1, endI);            //1 min avg of insp tidal volume in ml
    avgVti :=Mean50.Out_Mean;                       // Output of mean
    AOI_Mean50(Mean50, MVArr, 1, endI);             //1 min avg of minute ventilation in l/min
    avgMV :=Mean50.Out_Mean;                        // Output of mean
    AOI_Mean50(Mean50, CrsDynArr, 1, endI);         //1 min avg of dynamic respiratory system compliance in ml/cmH2O
    avgCrsDyn :=Mean50.Out_Mean;                        // Output of mean
    AOI_Mean50(Mean50, CrsQStatArr, 1, endI);       //1 min avg of quasistatic respiratory system compliance in ml/
cmH2O
    avgCrsQStat :=Mean50.Out_Mean;                      // Output of mean
    AOI_Mean50(Mean50, RawArr, 1, endI);            //1 min avg of airway resistance in cmH2O/l/s
    avgRaw :=Mean50.Out_Mean;                       // Output of mean
    CalcNewData :=0;
    AOI_FillArray_Real2D(FillLoop,FlowArrMem,0,2000);           // Clear all raw data
```

```
    AOI_FillArray_Real2D(FillLoop,FlowO2Mem ,0,2000);          // Clear all raw data
    AOI_FillArray_Real2D(FillLoop,PressArrMem,0,2000);         // Clear all raw data

end_if;
```

General

Type:	Structured Text	Number of Lines:	Original 232
In Program:	measCalc		

General

Configuration

Main:	Main	Inhibit program:	No
Fault:	None	Synchronize redundancy data after execution:	Yes

Monitor

Scan Times(Execution Time)

Max:	937 us	Last:	30 us

```
// Set MainFlow to start over
if StateMain = 0 then;
    StateMain := 10;
end_if;

IE_E := IE_counter +1;

// see if new data is valible from GUI
CsumData := FiO2+Vt + RRate + IE_I + IE_E + inspPauseTime + inspRampTime;

If CsumData <> CsumDataOld then;
    CsumDataOld := CsumData;
    NewSettings :=1;
    FiO2Running := FiO2;
    VtRunning := Vt;
    RRateRunning := RRate;
    IE_IRunning := IE_I;
    IE_ERunning := IE_E;
    inspPauseTimeRunning := inspPauseTime;
    inspRampTimeRunning := inspRampTime;
end_if;



// MainFlow sequence
case StateMain of


    // Run Setup
    10: if SetupDone then;                      // If setup is done
            StateMain := 20;                    // Set next stage
            SetupState := 0;

        else


            if SetupState = 0 then;
                SetupState := 10;
            end_if;                             // Start the setup program
        end_if;

    // Run CalcBreath
    20: if NewSettings then;                    // If changes has happend on the GUI

            if StateCalcBreath = 0 then;
                StateCalcBreath := 10;              // Start the CalcBreath setup
            end_if;
        else
            StateMain := 30;                    // Set next stage
        end_if;

    // Stop Ventilation
    30: if StopVent then;                                                   // Stop ventilation
            if Exp then
                if expCount
expiratory ends
                        > (expCountLength - 20) then;                       // Stop ventilatio 20*scanrate befor
                    VentOn :=0 ;                                            // Stop ventilation routine


                    AOI_FillArray_Real2D(FillLoop,FlowAir,0,2000);              // Clear all raw data
                    AOI_FillArray_Real2D(FillLoop,FlowO2,0,2000);               // Clear all raw data
                    AOI_FillArray_Real2D(FillLoop,PressArr,0,2000);             // Clear all raw data
```

```
                AOI_FillArray_Real2D(FillLoop,FlowArrMem,0,2000);           // Clear all raw data
                AOI_FillArray_Real2D(FillLoop,FlowO2Mem ,0,2000);           // Clear all raw data
                AOI_FillArray_Real2D(FillLoop,PressArrMem,0,2000);          // Clear all raw data
                AOI_FillArray_Real2D(FillLoop,_edbFlowArrFiltered,0,2000);  // Clear all raw data
                AOI_FillArray_Real2D(FillLoop,_edbFlowO2Filtered,0,2000);   // Clear all raw data
                AOI_FillArray_Real2D(FillLoop,_edbPressArrFiltered,0,2000); // Clear all raw data
                StopVent := 0;                                              // Reset the stop command
                Insp := 1;                                                  // Select inspiratory next start first
                Exp := 0;                                                   // Deselect expiratory next start
                expCount := 0;                                              // Reset the expiratory count
                StateMain := 0;                                             // Reset sequence and start over

            end_if;
        end_if;
    else
        StateMain := 40;                                    // Set next stage
    end_if;

// Run initVent
40: if StartVent then;                                      // If ventilator is started

    if StateInitVent = 0 then;
        StateInitVent := 10;                            // Start the initVent program
    end_if;

    else

        StateMain := 0;
        StateInitVent :=0;

    end_if;

end_case;
```

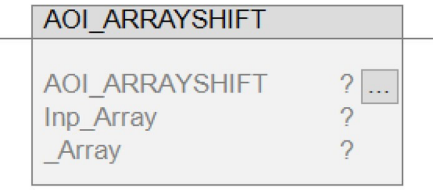
General

Type:	Structured Text	Number of Lines:	Original 91
In Program:	MainFlow		

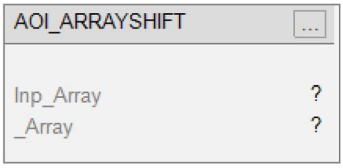
Signature Listing

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_ARRAYSHIFT(Inp_Array, _Array);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_ARRAYSHIFT	AOI_ARRAYSHIFT	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	Inp_Array	REAL	InOut	
X	_Array	REAL	InOut	

Extended Description

Execution

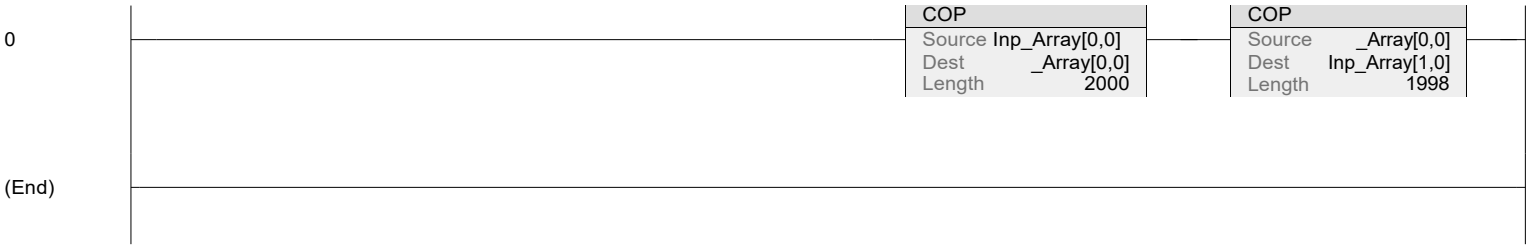
Condition	Description
EnableIn is true	

Revision v1.0 Notes

Name	Default	Data Type	Scope
Inp_Array		REAL[1000,2]	AOI_ARRAYSHIFT
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
*Inp_Array - AOI_ARRAYSHIFT/Logic - *0(COP), 0(COP)*			
_Array		REAL[1000,2]	AOI_ARRAYSHIFT
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
*_Array - AOI_ARRAYSHIFT/Logic - *0(COP), 0(COP)*			

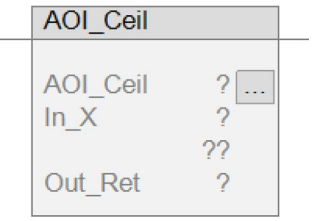
Name	Default	Data Type	Scope

No Tags Exist

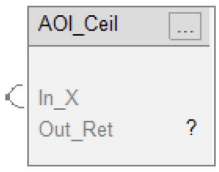

0
COP
Source Inp_Array[0,0]
Dest _Array[0,0]
Length 2000
COP
Source _Array[0,0]
Dest Inp_Array[1,0]
Length 1998
(End)

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_Ceil(In_X, Out_Ret);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_Ceil	AOI_Ceil	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	In_X	REAL	Input	
X	Out_Ret	REAL	InOut	

Extended Description

Execution

Condition	Description
EnableIn is true	

Revision v1.1 Notes

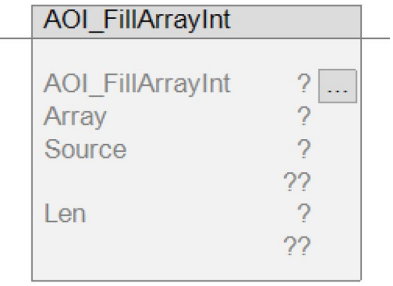
Name	Default	Data Type	Scope
In_X	0.0	REAL	AOI_Ceil
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_X - AOI_Ceil/Logic - #1, #3			
Out_Ret	??	REAL	AOI_Ceil
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
*Out_Ret - AOI_Ceil/Logic - *#6*			

Name	Default	Data Type	Scope
_aoiTrunc		AOI_Trunc	AOI_Ceil
Usage:	Local Tag		
External Access:	None		
*_aoiTrunc - AOI_Ceil/Logic - *#1*			
_aoiTrunc.EnableIn	1	BOOL	
Enable Input - System Defined Parameter			
_aoiTrunc.EnableOut	0	BOOL	
Enable Output - System Defined Parameter			
_n	0	DINT	AOI_Ceil
Usage:	Local Tag		
External Access:	None		
*_n - AOI_Ceil/Logic - #3, #4, #6, *#2, *#4*			
_trunc	0.0	REAL	AOI_Ceil
Usage:	Local Tag		
External Access:	None		
*_trunc - AOI_Ceil/Logic - #2, *#1*			

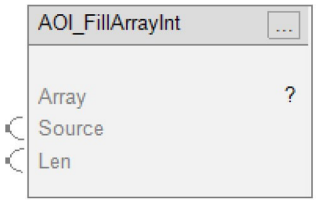
```
AOI_Trunc( _aoiTrunc, In_X, _trunc );
_n := _trunc;
if _n < In_X then
    _n := _n + 1;
end_if;
Out_Ret := _n;
```

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_FillArrayInt(Array, Source, Len);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_FillArrayInt	AOI_FillArrayInt	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	Array	INT	InOut	
X	Source	DINT	Input	
X	Len	DINT	Input	

Extended Description

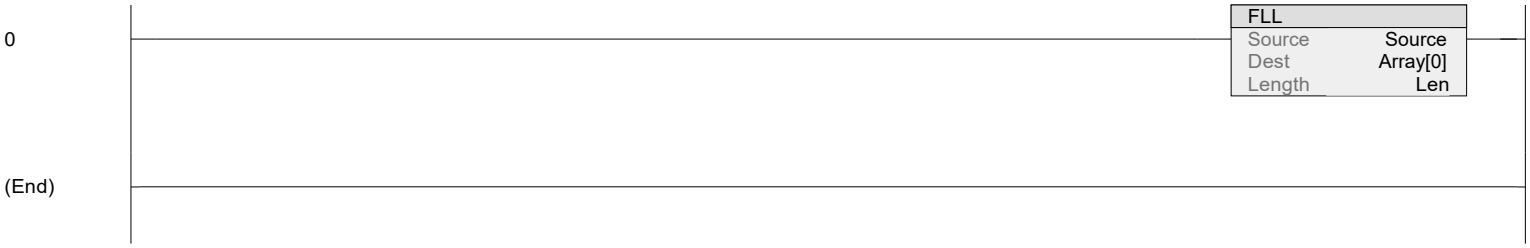
Execution

Condition	Description
EnableIn is true	

Revision v1.0 Notes

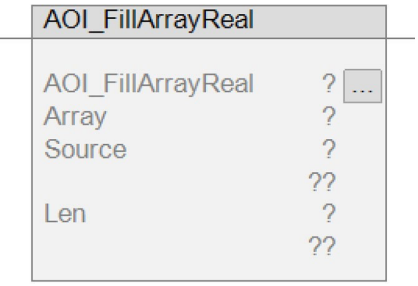
Name	Default	Data Type	Scope
Array		INT[1000]	AOI_FillArrayInt
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
*Array - AOI_FillArrayInt/Logic - *0(FLL)*			
Len	0	DINT	AOI_FillArrayInt
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
Len - AOI_FillArrayInt/Logic - 0(FLL)			
Source	0	DINT	AOI_FillArrayInt
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
Source - AOI_FillArrayInt/Logic - 0(FLL)			

Name	Default	Data Type	Scope
No Tags Exist			

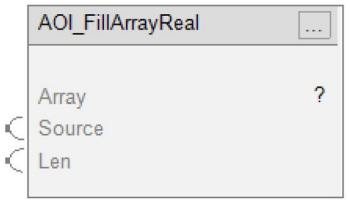

0
FLL
Source Source
Dest Array[0]
Length Len
(End)

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_FillArrayReal(Array, Source, Len);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_FillArrayReal	AOI_FillArrayReal	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	Array	REAL	InOut	
X	Source	DINT	Input	
X	Len	DINT	Input	

Extended Description

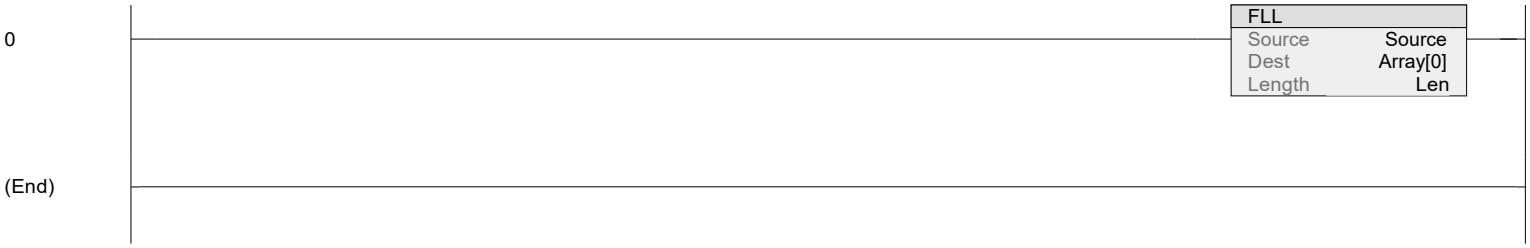
Execution

Condition	Description
EnableIn is true	

Revision v1.0 Notes

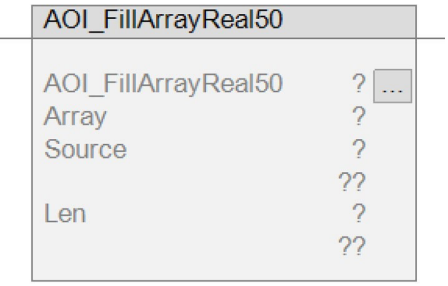
Name	Default	Data Type	Scope
Array		REAL[650]	AOI_FillArrayReal
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
*Array - AOI_FillArrayReal/Logic - *0(FLL)*			
Len	0	DINT	AOI_FillArrayReal
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
Len - AOI_FillArrayReal/Logic - 0(FLL)			
Source	0	DINT	AOI_FillArrayReal
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
Source - AOI_FillArrayReal/Logic - 0(FLL)			

Name	Default	Data Type	Scope
No Tags Exist			

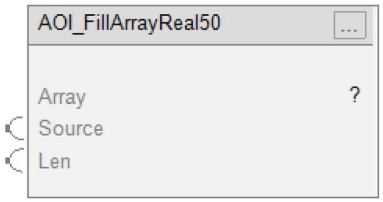

0
FLL
Source Source
Dest Array[0]
Length Len
(End)

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_FillArrayReal50(Array, Source, Len);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_FillArrayReal50	AOI_FillArrayReal50	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	Array	REAL	InOut	
X	Source	DINT	Input	
X	Len	DINT	Input	

Extended Description

Execution

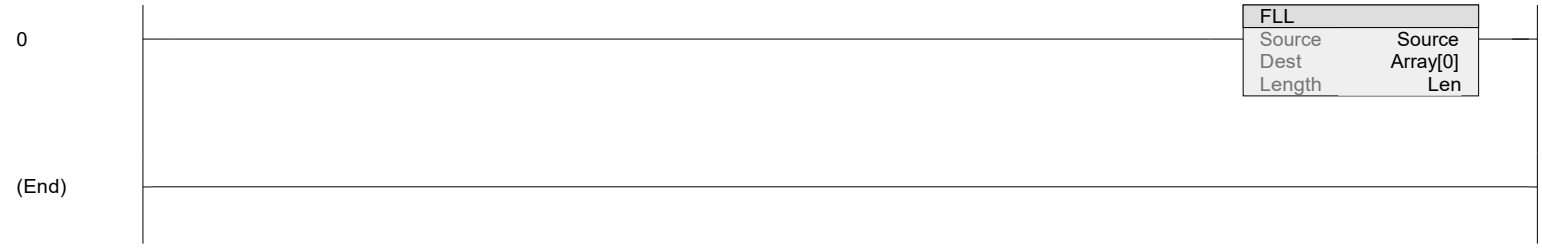
Condition	Description
EnableIn is true	

Revision v1.0 Notes

Name	Default	Data Type	Scope
Array		REAL[50]	AOI_FillArrayReal50
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
*Array - AOI_FillArrayReal50/Logic - *0(FLL)*			
Len	0	DINT	AOI_FillArrayReal50
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
Len - AOI_FillArrayReal50/Logic - 0(FLL)			
Source	0	DINT	AOI_FillArrayReal50
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
Source - AOI_FillArrayReal50/Logic - 0(FLL)			

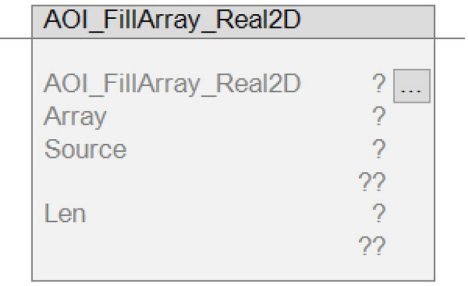
Name	Default	Data Type	Scope

No Tags Exist

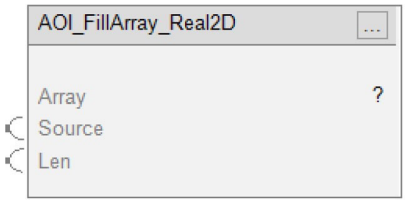

0
FLL
Source Source
Dest Array[0]
Length Len
(End)

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_FillArray_Real2D(Array, Source, Len);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_FillArray_Real2D	AOI_FillArray_Real2D	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	Array	REAL	InOut	
X	Source	DINT	Input	
X	Len	DINT	Input	

Extended Description

Execution

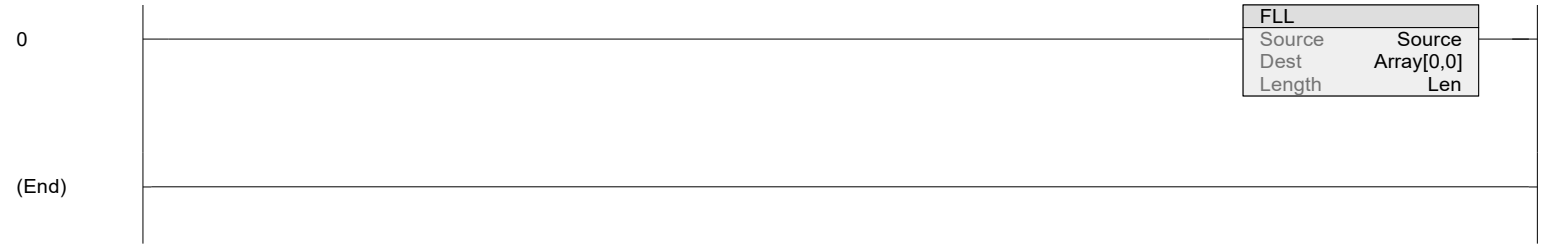
Condition	Description
EnableIn is true	

Revision v1.0 Notes

Name	Default	Data Type	Scope
Array		REAL[1000,2]	AOI_FillArray_Real2D
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
*Array - AOI_FillArray_Re l2D/Logic - *0(FLL)*			
Len	0	DINT	AOI_FillArray_Real2D
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
Len - AOI_FillArray_Real2D/Logic - 0(FLL)			
Source	0	DINT	AOI_FillArray_Real2D
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
Source - AOI_FillArray_Real2D/Logic - 0(FLL)			

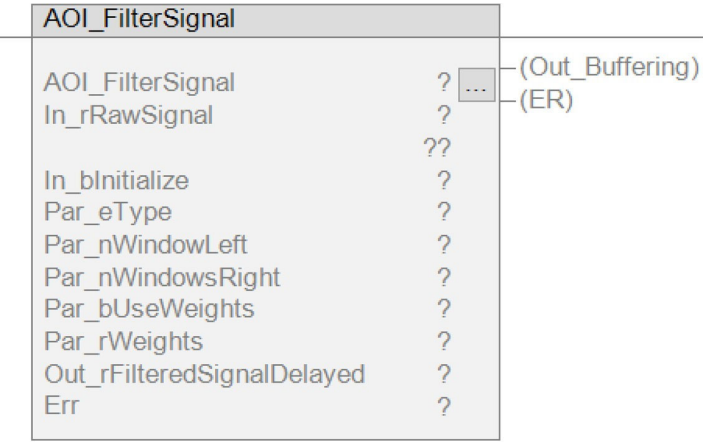
Name	Default	Data Type	Scope

No Tags Exist

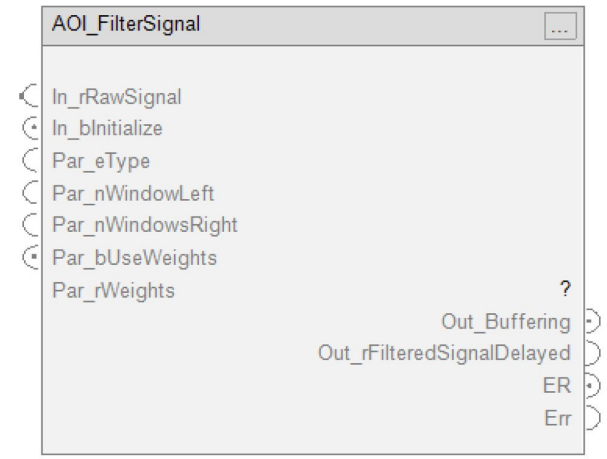

0
FLL
Source Source
Dest Array[0,0]
Length Len
(End)

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_FilterSignal(In_rRawSignal, Par_rWeights);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_FilterSignal	AOI_FilterSignal	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	In_rRawSignal	REAL	Input	
	In_bInitialize	BOOL	Input	
	Par_eType	DINT	Input	0: MEDIAN; 1: MEAN
	Par_nWindowLeft	DINT	Input	
	Par_nWindowsRight	DINT	Input	
	Par_bUseWeights	BOOL	Input	Used for mean filtering
X	Par_rWeights	REAL	InOut	Used for mean filtering
	Out_Buffering	BOOL	Output	
	Out_rFilteredSignalDelayed	REAL	Output	
	ER	BOOL	Output	
	Err	DINT	Output	

Extended Description

Execution

Condition	Description
EnableIn is true	
Prescan	

Revision v1.1 Notes

Name	Default	Data Type	Scope
ER	0	BOOL	AOI_FilterSignal
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read Only		
*ER - AOI_FilterSignal/Logic - #1, *#19, *#7*			
Err	0	DINT	AOI_FilterSignal
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*Err - AOI_FilterSignal/Logic - #18, *#14, *#15, *#16, *#17, *#8*			
In_bInitialize	0	BOOL	AOI_FilterSignal
Usage:	Input Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*In_bInitialize - AOI_FilterSignal/Logic - #1, #6, *#46*			
In_rRawSignal	0.0	REAL	AOI_FilterSignal
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_rRawSignal - AOI_FilterSignal/Logic - #49			
Out_Buffering	0	BOOL	AOI_FilterSignal
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*Out_Buffering - AOI_FilterSignal/Logic - #53, *#51*			
*Out_Buffering - AOI_FilterSignal/Prescan - *#14*			
Out_rFilteredSignalDelayed	0.0	REAL	AOI_FilterSignal
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read Only		
*Out_rFilteredSignalDelayed - AOI_FilterSignal/Logic - *#75, *#77, *#90*			
Par_bUseWeights	0	BOOL	AOI_FilterSignal
Used for mean filtering			
Usage:	Input Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
Par_bUseWeights - AOI_FilterSignal/Logic - #25			
Par_eType	0	DINT	AOI_FilterSignal
0: MEDIAN; 1: MEAN			
Usage:	Input Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
Par_eType - AOI_FilterSignal/Logic - #17, #17, #31			
Par_nWindowLeft	0	DINT	AOI_FilterSignal
Usage:	Input Parameter		
Required:	No		

Par_nWindowLeft (Continued)
Visible: Yes
External Access: Read/Write
Par_nWindowLeft - AOI_FilterSignal/Logic - #13, #14, #26

Par_nWindowsRight	0	DINT	AOI_FilterSignal

Usage: Input Parameter
Required: No
Visible: Yes
External Access: Read/Write
Par_nWindowsRight - AOI_FilterSignal/Logic - #13, #15, #27

Par_rWeights		REAL[1]	AOI_FilterSignal

Used for mean filtering
Usage: InOut Parameter
Required: Yes
Visible: Yes
Constant No
Par_rWeights - AOI_FilterSignal/Logic - #32
Par_rWeights - AOI_FilterSignal/Prescan - #7

Name	Default	Data Type	Scope
_bFirstScan	0	BOOL	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_bFirstScan - AOI_FilterSignal/Logic - #1, #6, *#45*			
*_bFirstScan - AOI_FilterSignal/Prescan - *#11*			
_bOdd	0	BOOL	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_bOdd - AOI_FilterSignal/Logic - #74, *#30*			
_bW	0	BOOL	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_bW - AOI_FilterSignal/Logic - #37, #87, *#25*			
_eType	0	DINT	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_eType - AOI_FilterSignal/Logic - #55, *#31*			
_i	0	DINT	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_i - AOI_FilterSignal/Logic - #40, #87, #88, *#39, *#86*			
_idx	0	DINT	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_idx - AOI_FilterSignal/Logic - #49, #88, #96, *#9, *#96*			
_n0	0	DINT	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_n0 - AOI_FilterSignal/Logic - *#26*			
_n1	0	DINT	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_n1 - AOI_FilterSignal/Logic - *#27*			
_n2	0	DINT	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_n2 - AOI_FilterSignal/Logic - #75, #77, #77, *#29*			
_nBufferUsage	0	DINT	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_nBufferUsage - AOI_FilterSignal/Logic - #50, #50, #51, *#10, *#50*			
_nM	0	DINT	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
_nM - AOI_FilterSignal/Logic - #16			
*_nM - AOI_FilterSignal/Prescan - #4, #6, #8, *#2, *#4, *#6, *#8*			
_nN	0	DINT	AOI_FilterSignal
Usage:	Local Tag		
External Access:	None		
*_nN - AOI_FilterSignal/Logic - #29, #30, #32, #34, #36, #39, #50, #51, #72, #86, #88, #96, *#28*			

_nTmp 0 DINT AOI_FilterSignal
Usage: Local Tag
External Access: None
*_nTmp - AOI_FilterSignal/Logic - #16, #28, *#13*
*_nTmp - AOI_FilterSignal/Prescan - #4, #4, #6, #6, #8, #8, *#3, *#5, *#7*

_rBufSorted REAL[31] AOI_FilterSignal
Usage: Local Tag
External Access: None
*_rBufSorted - AOI_FilterSignal/Logic - #75, #77, #77, *#73*
_rBufSorted - AOI_FilterSignal/Prescan - #5
*_rBufSorted[0] - AOI_FilterSignal/Logic - *#72*

_rRawBuf REAL[31] AOI_FilterSignal
Usage: Local Tag
External Access: None
*_rRawBuf - AOI_FilterSignal/Logic - #88, *#49*
_rRawBuf - AOI_FilterSignal/Prescan - #3
_rRawBuf[0] - AOI_FilterSignal/Logic - #72

_rSum 0.0 REAL AOI_FilterSignal
Usage: Local Tag
External Access: None
*_rSum - AOI_FilterSignal/Logic - #88, #90, *#85, *#88*

_rSumW 0.0 REAL AOI_FilterSignal
Usage: Local Tag
External Access: None
*_rSumW - AOI_FilterSignal/Logic - #40, #90, *#36, *#38, *#40*

_rWs REAL[31] AOI_FilterSignal
Usage: Local Tag
External Access: None
_rWs - AOI_FilterSignal/Logic - #40, #87
_rWs - AOI_FilterSignal/Prescan - #2
*_rWs[0] - AOI_FilterSignal/Logic - *#32*

_srt CONTROL AOI_FilterSignal
Usage: Local Tag
External Access: None
*_srt - AOI_FilterSignal/Logic - *#73*
_srt.LEN 0 DINT
*_srt.LEN - AOI_FilterSignal/Logic - *#34*
_srt.POS 0 DINT
*_srt.POS - AOI_FilterSignal/Logic - *#33*

_w 0.0 REAL AOI_FilterSignal
Usage: Local Tag
External Access: None
*_w - AOI_FilterSignal/Logic - #88, *#35, *#87*

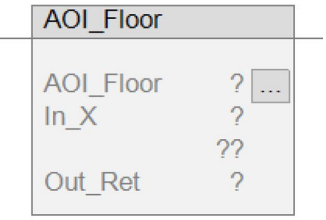
```
if ER and not ( _bFirstScan or In_bInitialize ) then
    tnd();
end_if;

// Initialization
if In_bInitialize or _bFirstScan then
    ER  := 0;
    Err := 0;
    _idx := 0;
    _nBufferUsage := 0;

    // Validate parameters
    _nTmp := Par_nWindowLeft + Par_nWindowsRight + 1;
    if Par_nWindowLeft   < 0 then Err := 1; end_if;
    if Par_nWindowsRight < 0 then Err := 2; end_if;
    if _nTmp > _nM  then Err := 3; end_if;
    if Par_eType <> 0 and Par_eType <> 1 then Err := 4; end_if;
    if Err <> 0 then
        ER := 1;
        tnd();
    end_if;

    // Store parameters ( make filter behaviour independent from outside
    //   parameters' changes ).
    _bW := Par_bUseWeights;
    _n0 := Par_nWindowLeft;
    _n1 := Par_nWindowsRight;
    _nN := _nTmp;
    _n2 := _nN / 2;
    _bOdd := ( _nN mod 2 ) = 1; // neighbourhood window composed of a odd number of points
    _eType := Par_eType;
    cop( Par_rWeights[0], _rWs[0], _nN );
    _srt.POS := 0;
    _srt.LEN := _nN;
    _w := 1.0;
    _rSumW := _nN;
    if _bW then
        _rSumW := 0.0;
        for _i := 0 to _nN - 1 do
            _rSumW := _rSumW + _rWs[_i];
        end_for;
    end_if;

end_if;
_bFirstScan := 0;
In_bInitialize := 0;

// While buffering data the filter isn't applied
_rRawBuf[ _idx ] := In_rRawSignal;
if _nBufferUsage < _nN then _nBufferUsage := _nBufferUsage + 1; end_if;
Out_Buffering := _nBufferUsage <> _nN;

if not Out_Buffering then

    case _eType of

        0: // MEDIAN
        // Very simple implementation but not very efficient.
```

```
        //   It's ok here because a single filtered value is
        //   calculated at every scan of the plc.
        // I'm using firmware instruction SRT to
        //   sort the buffer array, because it's faster
        //   than anything that could be implemented in
        //   structure text. In case SRT isn't available in
        //   other PLCs, a sorting algorithm could be implemented.
        //   For example, in file sort/sort.c of the Gnu Scientific
        //     Library (ftp://ftp.gnu.org/gnu/gsl/) there is an
        //     implementation of the heap sort algorithm.
        // TODO: replace the SRT instruction with such GLS
        //       implementation so that this code can be reused
        //       in any PLC.
        cop( _rRawBuf[0], _rBufSorted[0], _nN );
        SRT( _rBufSorted, 0, _srt );
        if _bOdd then
            Out_rFilteredSignalDelayed := _rBufSorted[_n2];
        else
            Out_rFilteredSignalDelayed := 0.5 * ( _rBufSorted[_n2-1] + _rBufSorted[_n2] );
        end_if;

        1: // MEAN
        // Note: instruction MAVE could be used here, but I find
        //       the following code clearer. Moreover the code
        //       can be used in any PLC which doesn't include the
        //       MAVE instruction.

        _rSum := 0.0;
        for _i := 0 to _nN - 1 do
            if _bW then _w := _rWs[_i]; end_if;
            _rSum := _rSum + _w * _rRawBuf[ ( _idx + 1 + _i ) mod _nN ];
        end_for;
        Out_rFilteredSignalDelayed := _rSum / _rSumW;

    end_case;

end_if;

_idx := ( _idx + 1 ) mod _nN;
```

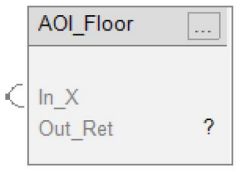
```
// Calulate _nN = minimum between size( Par_rWeights ), size(_rWs), size(_rBuf) and size(_rBufSorted)
size( _rWs, 0, _nM );
size( _rRawBuf, 0, _nTmp );
if _nTmp < _nM then _nM := _nTmp; end_if;
size( _rBufSorted, 0, _nTmp );
if _nTmp < _nM then _nM := _nTmp; end_if;
size( Par_rWeights, 0, _nTmp );
if _nTmp < _nM then _nM := _nTmp; end_if;

// Used for initialization
_bFirstScan := 1;

// Data must be buffered at the beginning
Out_Buffering := 1;
```

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_Floor(In_X, Out_Ret);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_Floor	AOI_Floor	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	In_X	REAL	Input	
X	Out_Ret	REAL	InOut	

Extended Description

Execution

Condition	Description
EnableIn is true	

Revision v1.1 Notes

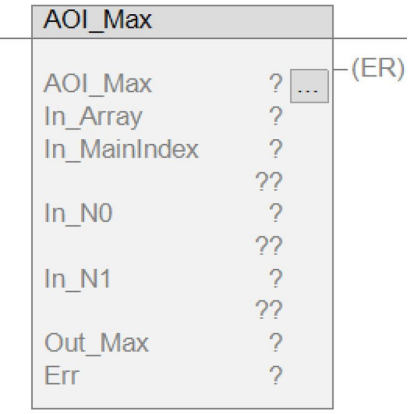
Name	Default	Data Type	Scope
In_X	0.0	REAL	AOI_Floor
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_X - AOI_Floor/Logic - #1, #2			
Out_Ret	??	REAL	AOI_Floor
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
*Out_Ret - AOI_Floor/Logic - *#7*			

Name	Default	Data Type	Scope
_n	0	DINT	AOI_Floor
Usage:	Local Tag		
External Access:	None		
*_n - AOI_Floor/Logic - #2 #3, #5, *#1*			
_ret	0.0	REAL	AOI_Floor
Usage:	Local Tag		
External Access:	None		
*_ret - AOI_Floor/Logic - #7, *#3, *#5*			

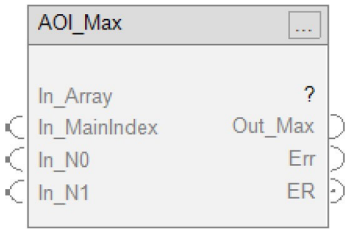
```
_n := In_X;
if In_X < _n then
    _ret := _n - 1;
else
    _ret := _n;
end_if;
Out_Ret := _ret;
```

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_Max(In_Array, In_MainIndex, In_N0, In_N1);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_Max	AOI_Max	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	In_Array	REAL	InOut	
X	In_MainIndex	DINT	Input	
X	In_N0	INT	Input	
X	In_N1	INT	Input	
	Out_Max	REAL	Output	
	Err	DINT	Output	
	ER	BOOL	Output	
	_i	DINT	Input	

Extended Description

Execution

Condition	Description
EnableIn is true	
Prescan	

Revision v1.0 Notes

Name	Default	Data Type	Scope
ER	0	BOOL	AOI_Max
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*ER - AOI_Max/Logic - *#1, *#10*			
Err	0	DINT	AOI_Max
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*Err - AOI_Max/Logic - #9, *#2, *#5, *#6, *#7, *#8*			
In_Array		REAL[1000,2]	AOI_Max
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
In_Array - AOI_Max/Logic - #15, #17			
In_Array - AOI_Max/Prescan - #1, #2			
In_MainIndex	0	DINT	AOI_Max
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_MainIndex - AOI_Max/Logic - #15, #17, #5, #5			
In_N0	0	INT	AOI_Max
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_N0 - AOI_Max/Logic - #15, #16, #6, #6, #8			
In_N1	0	INT	AOI_Max
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_N1 - AOI_Max/Logic - #16, #7, #7, #8			
Out_Max	0.0	REAL	AOI_Max
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*Out_Max - AOI_Max/Logic - *#20*			
_i	0	DINT	AOI_Max
Usage:	Input Parameter		
Required:	No		
Visible:	No		
External Access:	Read/Write		
*_i - AOI_Max/Logic - #17, *#16*			

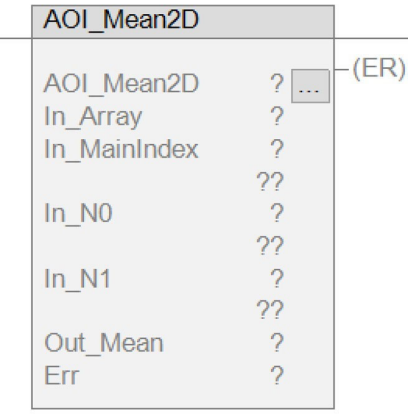
Name	Default	Data Type	Scope
_max	0.0	REAL	AOI_Max
Usage:	Local Tag		
External Access:	None		
*_max - AOI_Max/Logic - #18, #20, *#15, *#18*			
_maxDim0	0	DINT	AOI_Max
Usage:	Local Tag		
External Access:	None		
_maxDim0 - AOI_Max/Logic - #6, #7			
*_maxDim0 - AOI_Max/Prescan - *#1*			
_maxDim1	0	DINT	AOI_Max
Usage:	Local Tag		
External Access:	Read/Write		
_maxDim1 - AOI_Max/Logic - #5			
*_maxDim1 - AOI_Max/Prescan - *#2*			
_tmp	0.0	REAL	AOI_Max
Usage:	Local Tag		
External Access:	None		
*_tmp - AOI_Max/Logic - #18, #18, *#17*			

```
ER  := 0;
Err := 0;

// Some checks
if In_MainIndex < 0 or _maxDim1 <= In_MainIndex then Err := 1; end_if;
if In_N0 < 0        or _maxDim0 <= In_N0        then Err := 2; end_if;
if In_N1 < 0        or _maxDim0 <= In_N1        then Err := 3; end_if;
if In_N0 > In_N1                                then Err := 4; end_if;
if Err <> 0 then
    ER := 1;
    tnd();
end_if;

// Find maximum
_max := In_Array[ In_N0, In_MainIndex ];
for _i := In_N0 + 1 to In_N1 do
    _tmp := In_Array[ _i, In_MainIndex ];
    if _max < _tmp then _max := _tmp; end_if;
end_for;
Out_Max := _max;
```

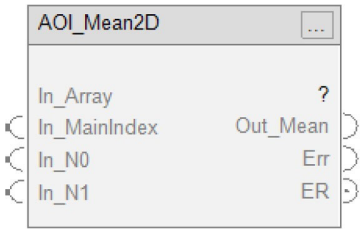
```
size( In_Array, 0, _maxDim0 ); // Dimension 0 is the most right index
size( In_Array, 1, _maxDim1 );
```

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_Mean2D(In_Array, In_MainIndex, In_N0, In_N1);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_Mean2D	AOI_Mean2D	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	In_Array	REAL	InOut	
X	In_MainIndex	DINT	Input	
X	In_N0	INT	Input	
X	In_N1	INT	Input	
	Out_Mean	REAL	Output	
	Err	DINT	Output	
	ER	BOOL	Output	
	_i	DINT	Input	

Extended Description

Execution

Condition	Description
EnableIn is true	
Prescan	

Revision v1.0 Notes

Name	Default	Data Type	Scope
ER	0	BOOL	AOI_Mean2D
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*ER - AOI_Mean2D/Logic - *#1, *#10*			
Err	0	DINT	AOI_Mean2D
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*Err - AOI_Mean2D/Logic - #9, *#2, *#5, *#6, *#7, *#8*			
In_Array		REAL[1000,2]	AOI_Mean2D
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
In_Array - AOI_Mean2D/Logic - #17			
In_Array - AOI_Mean2D/Prescan - #1, #2			
In_MainIndex	0	DINT	AOI_Mean2D
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_MainIndex - AOI_Mean2D/Logic - #17, #5, #5			
In_N0	0	INT	AOI_Mean2D
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_N0 - AOI_Mean2D/Logic - #16, #19, #6, #6, #8			
In_N1	0	INT	AOI_Mean2D
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_N1 - AOI_Mean2D/Logic - #16, #19, #7, #7, #8			
Out_Mean	0.0	REAL	AOI_Mean2D
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*Out_Mean - AOI_Mean2D/Logic - *#19*			
_i	0	DINT	AOI_Mean2D
Usage:	Input Parameter		
Required:	No		
Visible:	No		
External Access:	Read/Write		
*_i - AOI_Mean2D/Logic - #17, *#16*			

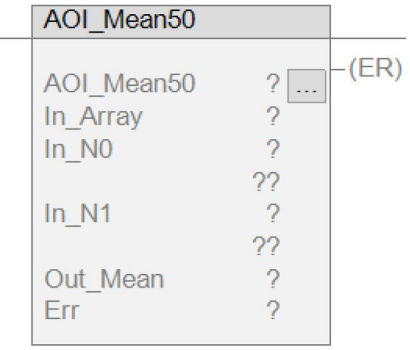
Name	Default	Data Type	Scope
_maxDim0	0	DINT	AOI_Mean2D
Usage:	Local Tag		
External Access:	None		
_maxDim0 - AOI_Mean2D/Logic - #6, #7			
*_maxDim0 - AOI_Mean2D/Prescan - *#1*			
_maxDim1	0	DINT	AOI_Mean2D
Usage:	Local Tag		
External Access:	Read/Write		
_maxDim1 - AOI_Mean2D/Logic - #5			
*_maxDim1 - AOI_Mean2D/Prescan - *#2*			
_sum	0.0	REAL	AOI_Mean2D
Usage:	Local Tag		
External Access:	None		
*_sum - AOI_Mean2D/Logic - #17, #19, *#15, *#17*			

```
ER  := 0;
Err := 0;

// Some checks
if In_MainIndex < 0 or _maxDim1 <= In_MainIndex then Err := 1; end_if;
if In_N0 < 0        or _maxDim0 <= In_N0        then Err := 2; end_if;
if In_N1 < 0        or _maxDim0 <= In_N1        then Err := 3; end_if;
if In_N0 > In_N1                                then Err := 4; end_if;
if Err <> 0 then
    ER := 1;
    tnd();
end_if;

// Find mean
_sum := 0.0;
for _i := In_N0 to In_N1 do
    _sum := _sum + In_Array[ _i, In_MainIndex ];
end_for;
Out_Mean
       := _sum / ( In_N1 - In_N0 + 1 );
```

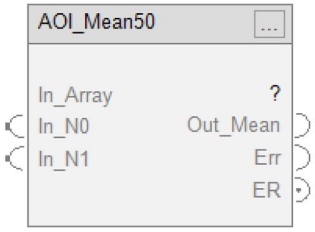
```
size( In_Array, 0, _maxDim0 ); // Dimension 0 is the most right index
size( In_Array, 1, _maxDim1 );
```

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_Mean50(In_Array, In_N0, In_N1);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_Mean50	AOI_Mean50	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	In_Array	REAL	InOut	
X	In_N0	INT	Input	
X	In_N1	INT	Input	
	Out_Mean	REAL	Output	
	Err	DINT	Output	
	ER	BOOL	Output	
	_i	DINT	Input	

Extended Description

Execution

Condition	Description
EnableIn is true	
Prescan	

Revision v1.0 Notes

Name	Default	Data Type	Scope
ER	0	BOOL	AOI_Mean50
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*ER - AOI_Mean50/Logic - *#1, *#9*			
Err	0	DINT	AOI_Mean50
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*Err - AOI_Mean50/Logic - #8, *#2, *#5, *#6, *#7*			
In_Array		REAL[50]	AOI_Mean50
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
In_Array - AOI_Mean50/Logic - #16			
In_Array - AOI_Mean50/Prescan - #1			
In_N0	0	INT	AOI_Mean50
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_N0 - AOI_Mean50/Logic - #15, #18, #5, #5, #7			
In_N1	0	INT	AOI_Mean50
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_N1 - AOI_Mean50/Logic - #15, #18, #6, #6, #7			
Out_Mean	0.0	REAL	AOI_Mean50
Usage:	Output Parameter		
Required:	No		
Visible:	Yes		
External Access:	Read/Write		
*Out_Mean - AOI_Mean50/Logic - *#18*			
_i	0	DINT	AOI_Mean50
Usage:	Input Parameter		
Required:	No		
Visible:	No		
External Access:	Read/Write		
*_i - AOI_Mean50/Logic - #16, *#15*			

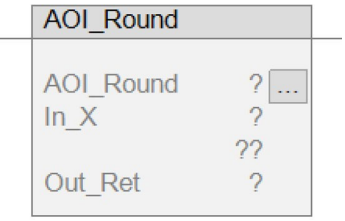
Name	Default	Data Type	Scope
_maxDim0	0	DINT	AOI_Mean50
Usage:	Local Tag		
External Access:	None		
_maxDim0 - AOI_Mean50/Logic - #5, #6			
*_maxDim0 - AOI_Mean50/Prescan - *#1*			
_sum	0.0	REAL	AOI_Mean50
Usage:	Local Tag		
External Access:	None		
*_sum - AOI_Mean50/Logic - #16, #18, *#14, *#16*			

```
ER  := 0;
Err := 0;

// Some checks
if In_N0 < 0        or _maxDim0 <= In_N0        then Err := 2; end_if;
if In_N1 < 0        or _maxDim0 <= In_N1        then Err := 3; end_if;
if In_N0 > In_N1                                then Err := 4; end_if;
if Err <> 0 then
    ER := 1;
    tnd();
end_if;

// Find mean
_sum := 0.0;
for _i := In_N0 to In_N1 do
    _sum := _sum + In_Array[ _i];
end_for;
Out_Mean
        := _sum / ( In_N1 - In_N0 + 1 );
```

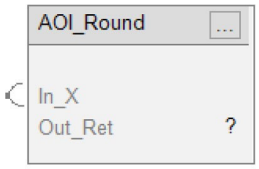
```
size( In_Array, 0, _maxDim0 ); // Dimension 0 is the most right index
```

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_Round(In_X, Out_Ret);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_Round	AOI_Round	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	In_X	REAL	Input	
X	Out_Ret	REAL	InOut	

Extended Description

Execution

Condition	Description
EnableIn is true	

Revision v1.0 Notes

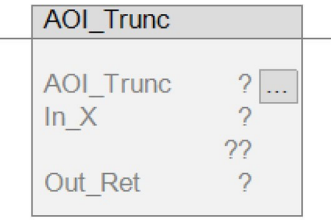
Name	Default	Data Type	Scope
In_X	0.0	REAL	AOI_Round
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_X - AOI_Round/Logic - #1			
Out_Ret	??	REAL	AOI_Round
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
*Out_Ret - AOI_Round/Logic - *#2*			

Name	Default	Data Type	Scope
_n	0	DINT	AOI_Round
Usage:	Local Tag		
External Access:	None		
*_n - AOI_Round/Logic - #2, *#1*			

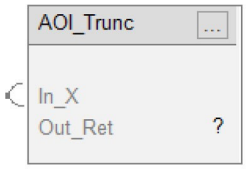
```
_n := In_X;
Out_Ret := _n;
```

Available Languages

Relay Ladder



Function Block



Structured Text

```
AOI_Trunc(In_X, Out_Ret);
```

Parameters

Required	Name	Data Type	Usage	Description
X	AOI_Trunc	AOI_Trunc	InOut	
	EnableIn	BOOL	Input	
	EnableOut	BOOL	Output	
X	In_X	REAL	Input	
X	Out_Ret	REAL	InOut	

Extended Description

Execution

Condition	Description
EnableIn is true	

Revision v1.0 Notes

Name	Default	Data Type	Scope
In_X	0.0	REAL	AOI_Trunc
Usage:	Input Parameter		
Required:	Yes		
Visible:	Yes		
External Access:	Read/Write		
In_X - AOI_Trunc/Logic - #7			
Out_Ret	??	REAL	AOI_Trunc
Usage:	InOut Parameter		
Required:	Yes		
Visible:	Yes		
Constant	No		
*Out_Ret - AOI_Trunc/Logic - *#7*			

Name	Default	Data Type	Scope
No Tags Exist			

```
// TODO: Logix has a truc function but other PLCs may not.
//       Replace the native function in Logix with an
//       implementation which will work with other PLC.
//       Note that AOI_Trunc and AOI_Ceil (which depends
//       on AOI_Trunc ) may not be needed in the project.

Out_Ret := trunc( In_X );
```

Data type Name: STRING

Description:

Size: 88 byte(s)

Name	Value	Data Type	Style
LEN		DINT	Decimal
External Access:	Read/Write		
DATA		SINT[82]	ASCII
External Access:	Read/Write		

Data type Name: AOI_ARRAYSHIFT

Description:

Size: 4 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		

Data type Name: AOI_Ceil

Description:

Size: 24 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
In_X		REAL	Float
External Access:	Read/Write		

Data type Name: AOI_FillArrayInt

Description:

Size: 12 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
Source		DINT	Decimal
External Access:	Read/Write		
Len		DINT	Decimal
External Access:	Read/Write		

Data type Name: AOI_FillArrayReal

Description:

Size: 12 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
Source		DINT	Decimal
External Access:	Read/Write		
Len		DINT	Decimal
External Access:	Read/Write		

Data type Name: AOI_FillArrayReal50

Description:

Size: 12 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
Source		DINT	Decimal
External Access:	Read/Write		
Len		DINT	Decimal
External Access:	Read/Write		

Data type Name: AOI_FillArray_Real2D

Description:

Size: 12 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
Source		DINT	Decimal
External Access:	Read/Write		
Len		DINT	Decimal
External Access:	Read/Write		

Data type Name: AOI_FilterSignal

Description:

Size: 468 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Pa ameter			
External Access:	Read Only		
In_rRawSignal		REAL	Float
External Access:	Read/Write		
In_bInitialize		BOOL	Decimal
External Access:	Read/Write		
Par_eType		DINT	Decimal
0: MEDIAN; 1: MEAN			
External Access:	Read/Write		
Par_nWindowLeft		DINT	Decimal
External Access:	Read/Write		
Par_nWindowsRight		DINT	Decimal
External Access:	Read/Write		
Par_bUseWeights		BOOL	Decimal
Used for mean filtering			
External Access:	Read/Write		
Out_Buffering		BOOL	Decimal
External Access:	Read/Write		
Out_rFilteredSignalDelayed		REAL	Float
External Access:	Read Only		
ER		BOOL	Decimal
External Access:	Read Only		
Err		DINT	Decimal
External Access:	Read/Write		

Data type Name: AOI_Floor

Description:

Size: 16 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
In_X		REAL	Float
External Access:	Read/Write		

Data type Name: AOI_Max

Description:

Size: 40 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
In_MainIndex		DINT	Decimal
External Access:	Read/Write		
In_N0		INT	Decimal
External Access:	Read/Write		
In_N1		INT	Decimal
External Access:	Read/Write		
Out_Max		REAL	Float
External Access:	Read/Write		
Err		DINT	Decimal
External Access:	Read/Write		
ER		BOOL	Decimal
External Access:	Read/Write		
_i		DINT	Decimal
External Access:	Read/Write		

Data type Name: AOI_Mean2D

Description:

Size: 36 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
In_MainIndex		DINT	Decimal
External Access:	Read/Write		
In_N0		INT	Decimal
External Access:	Read/Write		
In_N1		INT	Decimal
External Access:	Read/Write		
Out_Mean		REAL	Float
External Access:	Read/Write		
Err		DINT	Decimal
External Access:	Read/Write		
ER		BOOL	Decimal
External Access:	Read/Write		
_i		DINT	Decimal
External Access:	Read/Write		

Data type Name: AOI_Mean50

Description:

Size: 32 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
In_N0		INT	Decimal
External Access:	Read/Write		
In_N1		INT	Decimal
External Access:	Read/Write		
Out_Mean		REAL	Float
External Access:	Read/Write		
Err		DINT	Decimal
External Access:	Read/Write		
ER		BOOL	Decimal
External Access:	Read/Write		
_i		DINT	Decimal
External Access:	Read/Write		

Data type Name: AOI_Round

Description:

Size: 12 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
In_X		REAL	Float
External Access:	Read/Write		

Data type Name: AOI_Trunc

Description:

Size: 8 byte(s)

Name	Value	Data Type	Style
EnableIn		BOOL	Decimal
Enable Input - System Defined Parameter			
External Access:	Read Only		
EnableOut		BOOL	Decimal
Enable Output - System Defined Parameter			
External Access:	Read Only		
In_X		REAL	Float
External Access:	Read/Write		

1769 Bus : Local Modules

Local: [0] 1769-L27ERM-QBFC1B test

Type:	1769-L27ERM-QBFC1B CompactLogix™ 5370 Controller	Parent:	Local
Vendor:	Rockwell Automation/Allen-Bradley	Vendor ID:	1
Slot:	0	Electronic Keying:	Disabled
Revision:	32.13	Status:	Standby
Module Fault:	Offline	Inhibit Flag	Off

Embedded I/O : Local Modules

Local: [1] Embedded Discrete_IO

Type:	Embedded 16 Point 24V DC Sink/Source Input /16 Point 24V DC Source Output	Parent:	Local
Vendor:	Rockwell Automation/Allen-Bradley	Vendor ID:	1
Slot:	1	Electronic Keying:	Compatible Keying
Revision:	3.1	Status:	Standby
Module Fault:	Offline	Inhibit Flag	Off
Use Unicast:	No		

Module Defined Configuration Tag	Value	Data Type
Local:1:C		AB:Embedded_DiscreteIO1:C:0
.Filter0OffOn_0	0	BOOL
.Filter0OffOn_1	0	BOOL
.Filter0OffOn_2	0	BOOL
.Filter0OffOn_3	0	BOOL
.Filter0OnOff_4	0	BOOL
.Filter0OnOff_5	0	BOOL
.Filter0OnOff_6	0	BOOL
.Filter0OnOff_7	0	BOOL
.Filter1OffOn_0	0	BOOL
.Filter1OffOn_1	0	BOOL
.Filter1OffOn_2	0	BOOL
.Filter1OffOn_3	0	BOOL
.Filter1OnOff_4	0	BOOL
.Filter1OnOff_5	0	BOOL
.Filter1OnOff_6	0	BOOL
.Filter1OnOff_7	0	BOOL

Local: [2] Embedded Analog_IO

Type:	Embedded 4 Channel Universal Analog Input/2 Channel Analog Output	Parent:	Local
Vendor:	Rockwell Automation/Allen-Bradley	Vendor ID:	1
Slot:	2	Electronic Keying:	Compatible Keying
Revision:	1.1	Status:	Standby
Module Fault:	Offline	Inhibit Flag	Off
Use Unicast:	No		

Module Defined Configuration Tag	Value	Data Type
Local:2:C		AB:Embedded_AnalogIO1:C:0
.RealTimeSample	0	INT
.TimestampEn	0	BOOL
.Ch0InputFilter_0	1	BOOL
.Ch0InputFilter_1	0	BOOL
.Ch0InputFilter_2	1	BOOL
.Ch0InputFilter_3	0	BOOL

	1	BOOL
.Ch0DisableCJCompensation		
.Ch0OpenWire_5	0	BOOL
.Ch0OpenWire_6	0	BOOL
.Ch0OpenWireEn	0	BOOL
.Ch0AlarmLatchEn	0	BOOL
.Ch0AlarmEn	1	BOOL
.Ch0InputEn	1	BOOL
.Ch0InputRange	4	SINT
	1	BOOL
.Ch0InputDataFormat_0		
	0	BOOL
.Ch0InputDataFormat_1		
	0	BOOL
.Ch0InputDataFormat_2		
.Ch0TempUnits	0	BOOL
	0	BOOL
.Ch0RTD_ResistanceMeasType_6		
	0	BOOL
.Ch0RTD_ResistanceMeasType_7		
.Ch0HAlarmLimit	2000	INT
.Ch0LAlarmLimit	500	INT
	0	INT
.Ch0AlarmDeadband		
.Ch1InputFilter_0	1	BOOL
.Ch1InputFilter_1	0	BOOL
.Ch1InputFilter_2	1	BOOL
.Ch1InputFilter_3	0	BOOL
	1	BOOL
.Ch1DisableCJCompensation		
.Ch1OpenWire_5	0	BOOL
.Ch1OpenWire_6	0	BOOL
.Ch1OpenWireEn	0	BOOL
.Ch1AlarmLatchEn	0	BOOL
.Ch1AlarmEn	1	BOOL
.Ch1InputEn	1	BOOL
.Ch1InputRange	4	SINT
	1	BOOL
.Ch1InputDataFormat_0		
	0	BOOL
.Ch1InputDataFormat_1		
	0	BOOL
.Ch1InputDataFormat_2		
.Ch1TempUnits	0	BOOL
.Ch1HAlarmLimit	5250	INT
.Ch1LAlarmLimit	500	INT
	0	INT
.Ch1AlarmDeadband		
.Ch2InputFilter_0	1	BOOL
.Ch2InputFilter_1	0	BOOL
.Ch2InputFilter_2	1	BOOL
.Ch2InputFilter_3	0	BOOL

	1	BOOL
.Ch2DisableCJCompensation		
.Ch2OpenWire_5	0	BOOL
.Ch2OpenWire_6	0	BOOL
.Ch2OpenWireEn	0	BOOL
.Ch2AlarmLatchEn	1	BOOL
.Ch2AlarmEn	1	BOOL
.Ch2InputEn	1	BOOL
.Ch2InputRange	3	SINT
	1	BOOL
.Ch2InputDataFormat_0		
	0	BOOL
.Ch2InputDataFormat_1		
	0	BOOL
.Ch2InputDataFormat_2		
.Ch2TempUnits	0	BOOL
	0	BOOL
.Ch2RTD_ResistanceMeasType_6		
	0	BOOL
.Ch2RTD_ResistanceMeasType_7		
.Ch2HAlarmLimit	5600	INT
.Ch2LAlarmLimit	3200	INT
	0	INT
.Ch2AlarmDeadband		
.Ch3InputFilter_0	1	BOOL
.Ch3InputFilter_1	0	BOOL
.Ch3InputFilter_2	1	BOOL
.Ch3InputFilter_3	0	BOOL
	1	BOOL
.Ch3DisableCJCompensation		
.Ch3OpenWire_5	0	BOOL
.Ch3OpenWire_6	0	BOOL
.Ch3OpenWireEn	0	BOOL
.Ch3AlarmLatchEn	0	BOOL
.Ch3AlarmEn	0	BOOL
.Ch3InputEn	1	BOOL
.Ch3InputRange	1	SINT
	1	BOOL
.Ch3InputDataFormat_0		
	0	BOOL
.Ch3InputDataFormat_1		
	0	BOOL
.Ch3InputDataFormat_2		
.Ch3TempUnits	0	BOOL
.Ch3HAlarmLimit	2	INT
.Ch3LAlarmLimit	0	INT
	0	INT
.Ch3AlarmDeadband		
.CJTempUnits	0	BOOL
.CJWeightedProfile	0	BOOL
	0	BOOL
.CyclicCalibrationEn		

	0	BOOL
.UpdateCJCompensationEn		
	0	BOOL
.Ch0LimitAlarmLatchEn		
.Ch0OutputEn	1	BOOL
	3	SINT
.Ch0OutputRangeType		
	1	SINT
.Ch0OutputDataFormat		
.Ch0HClampValue	21000	INT
.Ch0LClampValue	3200	INT
	0	BOOL
.Ch1LimitAlarmLatchEn		
.Ch1OutputEn	1	BOOL
	3	SINT
.Ch1OutputRangeType		
	1	SINT
.Ch1OutputDataFormat		
.Ch1HClampValue	21000	INT
.Ch1LClampValue	3200	INT

Local: [3] Embedded Counters

Type:	Embedded 4 Channel High Speed Counter	Parent:	Local
Vendor:	Rockwell Automation/Allen-Bradley	Vendor ID:	1
Slot:	3	Electronic Keying:	Compatible Keying
Revision:	1.1	Status:	Standby
Module Fault:	Offline	Inhibit Flag	Off
Use Unicast:	No		

Module Defined Configuration Tag	Value	Data Type
Local:3:C		AB:Embedded_HSC1:C:0
.Config0	2#0000_0001_0000_0000	INT
	0	BOOL
.OverCurrentLatchOff		
.CtrReset	0	BOOL
.ProgToFaultEn	0	BOOL
	1	BOOL
.NumberOfCounters_0		
	0	BOOL
.NumberOfCounters_1		
.Ctr0ResetEn	0	BOOL
.Ctr1ResetEn	0	BOOL
.Ctr2ResetEn	0	BOOL
.Ctr3ResetEn	0	BOOL
.Filter	2#0000_0000_0000_0000	INT
.FilterA0_0	0	BOOL
.FilterA0_1	0	BOOL
.FilterB0_0	0	BOOL
.FilterB0_1	0	BOOL
.FilterZ0_0	0	BOOL
.FilterZ0_1	0	BOOL
.FilterA1_0	0	BOOL

.FilterA1_1	0	BOOL
.FilterB1_0	0	BOOL
.FilterB1_1	0	BOOL
.FilterZ1_0	0	BOOL
.FilterZ1_1	0	BOOL
.Ctr0MaxCount	2147483647	DINT
.Ctr0MinCount	-2147483648	DINT
.Ctr0Preset	0	DINT
.Ctr0Hysteresis	0	INT
.Ctr0Scaler	1	INT
	10	INT
.Ctr0CyclicRateUpdateTime		
.Ctr0Config	2#0000_0000	SINT
	0	BOOL
.Ctr0OperationMode_0		
	0	BOOL
.Ctr0OperationMode_1		
	0	BOOL
.Ctr0OperationMode_2		
	0	BOOL
.Ctr0StoreOnRisingZ		
.Ctr0HoldOnZ	0	BOOL
	0	BOOL
.Ctr0PresetOnRisingZ		
.Ctr0Linear	0	BOOL
.Ctr1MaxCount	2147483647	DINT
.Ctr1MinCount	-2147483648	DINT
.Ctr1Preset	0	DINT
.Ctr1Hysteresis	0	INT
.Ctr1Scaler	1	INT
	10	INT
.Ctr1CyclicRateUpdateTime		
.Ctr1Config	2#0000_0000	SINT
	0	BOOL
.Ctr1OperationMode_0		
	0	BOOL
.Ctr1OperationMode_1		
	0	BOOL
.Ctr1OperationMode_2		
	0	BOOL
.Ctr1StoreOnRisingZ		
.Ctr1HoldOnZ	0	BOOL
	0	BOOL
.Ctr1PresetOnRisingZ		
.Ctr1Linear	0	BOOL
.Ctr2MaxCount	0	DINT
.Ctr2MinCount	0	DINT
.Ctr2Preset	0	DINT
.Ctr2Hysteresis	0	INT
.Ctr2Scaler	0	INT
	0	INT
.Ctr2CyclicRateUpdateTime		
.Ctr2Linear	0	BOOL
.Ctr3MaxCount	0	DINT
.Ctr3MinCount	0	DINT

.Ctr3Preset	0	DINT
.Ctr3Hysteresis	0	INT
.Ctr3Scaler	0	INT
	0	INT
.Ctr3CyclicRateUpdateTime		
.Ctr3Linear	0	BOOL
.Range0To11		AB:1769_HSC1_Range:C:0[12]